МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно-космічної техніки

Кафедра вищої математики та системного аналізу

Пояснювальна записка до дипломної роботи

магістра
(освітньо-кваліфікаційний рівень)

на тему «Автоматизована система аналізу антропометричних параметрів»

ХАІ.405.463-м.124.1504010.20О

Виконав: студент 6 курсу групи 463-м
спеціальності 124 «Системний аналіз»
освітньої програми «Системний аналіз і управління»
Булгакова Є.В.
(прізвище та ініціали студента)
Керівник: Макарічев В.О.
(прізвище та ініціали)
Рецензент: Колодяжний В.М.
(прізвище та ініціали)

Харків – 2020

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно – космічної техніки
Кафедра 405 «Вищої математики та системного аналізу»
Рівень вищої освіти другий (магістрський)
Спеціальність (напрям підготовки) 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор, д. ф-м. н. Ніколаєв О.Г.

_____ О.Г. Ніколаєв
(підпис) (ініціали та прізвище)
«___» _____ 2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Булгакової Євгенії Володимирівни
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Автоматизована система аналізу антропометричних параметрів
керівник роботи канд. фіз.-мат. наук, доцент Макарічев Віктор Олександрович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету № 1819-уч від « 03 » 11 2020р.
2. Термін подання студентом проекту (роботи) « » _____20___р.
3. Вихідні дані до проекту (роботи) створення програмного продукту
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Опис автоматизованої системи аналізу антропометричних параментрів людини з точки зору системного аналізу.
2. Модельний вигляд об'єкта дослідження.
3. Вибір платформи та середи розробки программного продукту.
4. Розробка програмного продукту.
5. Визначення розрахунку собівартості та ціни програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____
_____
_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Загальна частина	канд. фіз.-мат. наук, доцент каф. 405 Макарічев В.О.		
Економічна частина	ст.викладач каф. 601 Зіброва О.В.		

Нормоконтроль _____ _____ « » _____20___р.
(підпис) (прізвище та ініціали)

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд апріорної інформації по заданій тематиці		
2	Визначення актуальності проблеми, цілей, задач і методів дослідження		
3	Опис автоматизованої системи аналізу антропометричних параметрів людини з точки зору системного аналізу		
4	Модельний вигляд об’єкта дослідження		
5	Вибір платформи та середовища розробки программного продукту		
6	Розробка програмного продукту		
7	Визначення розрахунку собівартості програмного продукту		
8	Оформлення пояснювальної записки		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи містить: 97 с., 11 рис., 15 табл., 38 джерел посилання.

АВТОМАТИЗАЦІЯ, АНТРОПОМЕТРІЯ, АНТРОПОМЕТРИЧНІ ПАРАМЕТРИ, ІНТЕРНЕТ РЕЧЕЙ, СИСТЕМНИЙ АНАЛІЗ.

Об'єкт дослідження: система антропометричних параметрів.

Мета роботи: розробити автоматизовану систему аналізу антропометричних параметрів.

Предмет дослідження: антропометричні параметри; методи аналізу антропометричних параметрів; структура системи аналізу антропометричних параметрів.

Основні завдання: провести аналіз антропометричних параметрів щодо їх вхідних і вихідних параметрів; сформувати основні вимоги та рекомендації для розробки автоматизованої системи аналізу антропометричних параметрів.

Методи дослідження: методи системного аналізу; методи та засоби комп'ютерних інформаційних технологій.

Доводиться системність об'єкта дослідження, проводиться структурний, функціональний, інформаційний та класифікаційний опис. Визначаються основні вхідні та вихідні параметри. Розглядаються існуючі системи аналізу антропометричних параметрів.

РЕФЕРАТ

Пояснительная записка к дипломной работе содержит: 97 с., 11 рис., 15 табл., 38 источников информации.

АВТОМАТИЗАЦИЯ, АНТРОПОМЕТРИЯ, АНТРОПОМЕТРИЧЕСКИЕ ДАННЫЕ, ИНТЕРНЕТ ВЕЩЕЙ, СИСТЕМНЫЙ АНАЛИЗ.

Объект исследования: система антропометрических данных.

Цель работы: разработать автоматизированную систему анализа антропометрических данных.

Предмет исследования: антропометрические данные; методы анализа антропометрических данных; структура системы анализа антропометрических данных.

Основные задачи: провести анализ антропометрических данных об их входных и выходных параметров; сформировать основные требования и рекомендации для разработки автоматизированной системы анализа антропометрических данных.

Методы исследования: методы системного анализа; методы и средства компьютерных информационных технологий.

Доказывается системность объекта исследования, проводится структурное, функциональное, информационное и классификационное описание. Определяются основные входные и выходные параметры. Рассматриваются существующие системы анализа антропометрических данных.

ABSTRACT

Explanatory note to the thesis contains: 97 p., 11 fig., 15 tabl., 38 sources of information.

AUTOMATION, ANALYSIS, ANTHROPOMETRY, ANTROPOMETRIC DATA, INTERNET OF THINGS, SYSTEM ANALYSIS.

Object is the system of anthropometric data.

The purpose of this research: develop an automated system for analyzing anthropometric data.

The subject of research: anthropometric data; methods of analysis of anthropometric data; structure of anthropometric data analysis system.

The main task: to conduct analysis of anthropometric data on their input and output parameters; to formulate the basic requirements and recommendations for the development of an automated system for analyzing anthropometric data.

Methods: methods of system analysis; methods and tools of computer information technology.

The system of the research object is proved, structural, functional, information and classification description is carried out. The main input and output parameters are determined. The existing systems of analysis of anthropometric data are considered.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БД – бази параметрів.

ЄСВ – єдиний соціальний внесок.

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

ШІ – штучний інтелект.

MVC – Model-view-controller.

OS – operating system.

IoT – internet of things.

ВСТУП

Актуальність обраної теми зараз досить велика, так як якісний аналіз антропометричних параметрів дозволить мотивувати людей на ведення здорового способу життя і скоротити захворюваність і смертність людства.

Як об'єкт дослідження було обрано систему антропометричних параметрів.

Метою даної дипломної роботи є розробка автоматизованої системи аналізу антропометричних параметрів.

Відповідно до мети дипломної роботи були поставлені наступні завдання: провести аналіз антропометричних параметрів щодо їх вхідних і вихідних параметрів; сформувати основні вимоги та рекомендації для розробки автоматизованої системи аналізу антропометричних параметрів.

Аналіз досліджень і публікацій [23, 26] показав, що в реальному світі люди турбуються і піклуються про своє здоров'я і прагнуть до його поліпшення. Дослідження, яке проводиться в цій роботі, допоможе людству вирішити проблеми зі здоров'ям.

У роботі відзначається, що в майбутньому варто приділити увагу удосконаленню даної системи аналізу антропометричних параметрів для її більш точної роботи.

У першому розділі даної дипломної роботи представлена інформація про актуальність даної теми, порівняння недоліків і переваг аналогічних систем, визначені об'єкт, предмет, мета, завдання та методи дослідження.

У другому розділі проведено морфологічний, функціональний і інформаційний аналіз досліджуваного об'єкта. А також дослідження класифікації системи.

У третьому розділі описана модель об'єкта дослідження і його рівняння, наведено опис елементів моделі і визначені вхідні та вихідні величини даного об'єкта.

У четвертому розділі обґрунтований вибір середовища розробки програмного забезпечення, описан процес розробки.

У п'ятому розділі проводиться розрахунок економічної складової програмного продукту.

У висновку описані результати наведеної роботи.

1 СИСТЕМА АНТРОПОМЕТРИЧНИХ ПАРАМЕТРІВ

1.1 Визначення та основні поняття

Антропометрія – один з основних методів антропологічного дослідження, який полягає у вимірюванні тіла людини і його частин з метою встановлення вікових, статевих, расових та інших особливостей фізичної будови, що дозволяє дати кількісну характеристику їх мінливості [4, 24, 29].

Антропометричні дані – величина показників тіла, виміряних в умовах відносної нерухомості людини. Під цим поняттям можна об'єднати всі статичні параметри, як всього організму в цілому (зріст, вага), так і окремих його частин (окружність голови, довжина руки, розміри стопи і так далі). Роль антропометричних параметрів досить велика. Завдяки статистичним дослідженням вдалося встановити параметри норми для людей різного віку, статі та навіть расової приналежності. Відхилення від них в одних випадках є особливістю самої людини, а в інших може свідчити про серйозні захворювання [5, 28].

Під аналізом антропометричних параметрів будемо розуміти визначення стану здоров'я людини на підставі антропометричних показників. [1].

Залежно від об'єкта дослідження розрізняють [Словник, с.]:

• соматометрію (вимір живої людини),

• краніометрії (вимір черепа),

• остеометрію (вимір кісток скелета) [2].

До антропометрії відносять також антропоскопію - якісну (описову) характеристику форм частин тіла, форми голови, рис обличчя, пігментації шкіри, волосся, райдужної оболонки очей, зовнішній огляд, визначення щільності і складу маси тіла, визначення абсолютної м'язової маси, сили м'язів, вимір гнучкості і рухливості, сили і витривалості, тести і оцінки силових показників і рухливості, а також визначення вмісту води в масі тіла [6, 30].

Головними антропометричними даними є зріст і вага. Саме вони найчастіше використовуються в сучасній медицині. На підставі вже тільки цих двох показників можна розрахувати, чи є у людини проблеми зі здоров'ям [22].

У представленій роботі були поділені параметри для антропометричні дослідження на такі класи:

• зовнішні параметри (ім'я, зріст, вага, стать, вік);

• внутрішні параметри (тривалість сну, наявність шкідливих звичок, фізична активність)

І для дослідження будуть використовуватися тільки дані параметри, так як вони вважаються більш точними для аналізу і прогнозування здоров'я сучасної людини [10].

1.2 Огляд літератури та інших інформаційних джерел

Огляд досліджень і публікацій виявив, що аналоги даного дослідження існують, але вони мають ряд недоліків, таких як: використання не достатньої кількості параметрів для вирішення необхідних завдань, відсутність можливостей повного аналізу якості життя і стану здоров'я людини [7-9]. Наприклад:

• додаток, прогнозуючий головний біль – «My pain Diary» (платне, тільки для користувачів Android);

• додаток, що фіксує болі у тілі, по типу щоденника болю – «CatchMyPain» (безкоштовне, багатоформений додаток);

• додаток, що дозволяє легко відстежувати біль та симптоми користувача й доповідати лікарю – «My pain Diary: Gold Edition» (платне, тільки для користувачів IOS);

• ШІ, який прогнозує смерть пацієнтів по їх історії хвороб («Google»).

У результаті виявлення проблем, які представлені у вигляді дерева проблем на рисунку 1.1, що виникають при аналізі автоматизованих систем аналізу антропометричних параметрів, можна виділити 3 цілі.

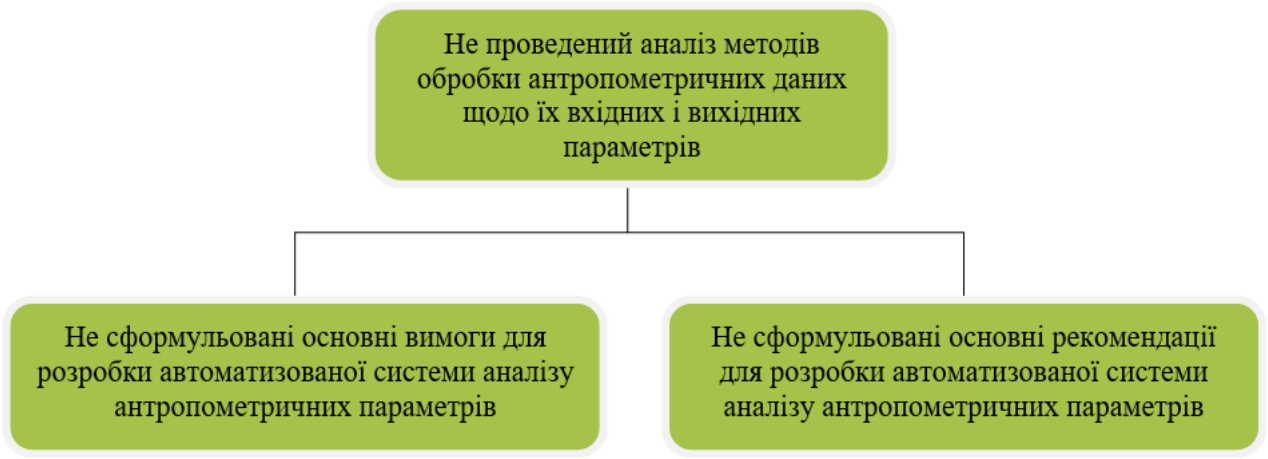

Рисунок 1.1 – Дерево проблем

Такі як:

• провести аналіз методів обробки антропометричних параметрів щодо вхідних і вихідних параметрів

• сформулювати основні вимоги для розробки автоматизованої системи аналізу антропометричних параметрів

• сформулювати основні рекомендації для розробки автоматизованої системи антропометричних параметрів

Такі цілі можна уявити як дерево цілей, наданого на рисунку 1.2.

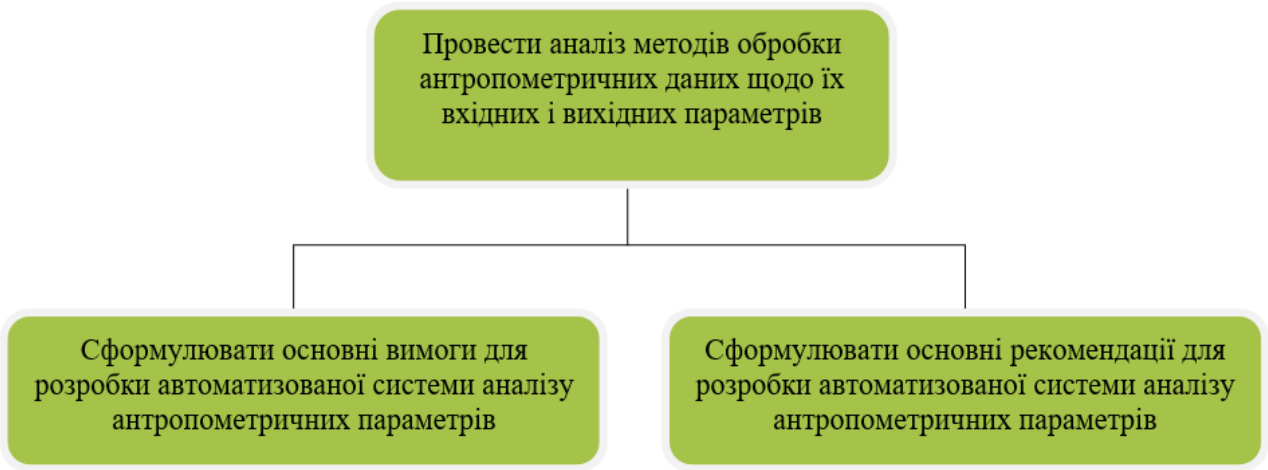

Рисунок 1.2 – Дерево цілей

1.3 Актуальність та особливості дослідження

Актуальність завдання можливостей створення автоматизованої системи аналізу антропометричних параметрів визначається тим, що розробка такої системи робить фактично доступним використання методів антропометрії для будь-якого користувача [3, 19-21, 25]. А це впливає на такі важливі чинники як:

1. Аналіз якості життя людини.
2. Визначення мотивації особистості.
3. Прогнозування стану здоров'я людини.
4. Попередження можливості захворювання.

Все це дозволяє поліпшити життя людей і скоротити смертність на нашій планеті.

Особливостями автоматизованої системи має бути те, що вона:

• розрахована на будь-яку людину;

• дозволить самостійно через спілкування людини з нею оцінити якість життя самої особистості;

• дозволить попередити користувача автоматизованої системи про можливі проблеми зі здоров'ям.

1.4 Мети й завдання дослідження

На основі розглянутих інформаційних джерел були складені об'єкт, предмет, мета і завдання дослідження.

Об'єкт дослідження: система антропометричних параметрів.

Предмет дослідження:

• антропометричні параметри

• методи аналізу антропометричних параметрів

• структура системи аналізу антропометричних параметрів

Мета дослідження: розробка автоматизованої системи аналізу антропометричних параметрів.

Завдання дослідження:

• провести аналіз антропометричних параметрів щодо їх вхідних і вихідних параметрів

• сформувати основні вимоги та рекомендації для розробки автоматизованої системи аналізу антропометричних параметрів

Методи дослідження:

• методи системного аналізу

• методи та засоби комп'ютерних інформаційних технологій

2 ОПИС СИСТЕМИ АНТРОПОМЕТРИЧНИХ ПАРАМЕТРІВ З ТОЧКИ ЗОРУ СИСТЕМНОГО АНАЛІЗУ

2.1 Морфологічний опис

Структурний опис системи антропометричних параметрів, виконано на підставі аналізу інформаційних джерел [11-13,28].

З цих джерел слід:

1) антропометричні дослідження починається з формування вхідних параметрів;

2) на підставі вхідних параметрів виділяються внутрішні і зовнішні параметри;

3) аналіз системи антропометричних параметрів працює на основі двох модулів - обробки та аналізу;

4) параметри на виході формуються за допомогою модуля видачі результатів.

Приведена схема роботи дозволила уявити систему в морфологічному вигляді.

Склад системи наступний:

1. Система антропометричних параметрів

1.1. Система формування вхідних параметрів

1.1.1. Підсистема виділення зовнішніх параметрів

1.1.2. Підсистема виділення внутрішніх параметрів

1.2. Система аналізу антропометричних параметрів

1.2.1.Модуль обробки вхідних параметрів

1.2.2.Модуль аналізу оброблених параметрів

1.3. Модуль видачі результатів

При цьому представлена система є системою з точки зору системного аналізу, так як вона характеризується такими властивостями:

Емерджентність: розглянута система має властивість емерджентності, оскільки в цілому система виробляє збір параметрів, аналіз і обробку параметрів. Таким властивістю не володіє будь-який окремий елемент цієї системи.

Цілісність: система є цілісною. Наприклад, при відсутності модуля обробки буде порушений аналіз системи і це призведе до непрацездатності всієї системи.

Адитивність: система має властивість адитивності, оскільки властивість системи антропометричних параметрів представляється як сума властивостей елементів системи антропометричних параметрів.

Синергізм: досліджувана система має властивість синергізму, оскільки якісна робота системи аналізу антропометричних параметрів замотівірует і поліпшить стан здоров'я людства.

Прогресуюча факторизація: характеризується тим, що всі елементи залежать один від одного, так як отримати оптимальний результат можна тільки якщо об'єднати всі елементи системи.

Ізоморфізм: дана система має властивості ізоморфізму, так як елементи системи мають подібні властивості. Наприклад, зовнішні і внутрішні параметри мають подібні властивості.

Тип елементного складу: змішаний тип.

Тип зв'язків: інформаційний, так як зв'язку переносять інформацію між елементами.

Тип структури: дана система має ієрархічною структурою.

2.2 Функціональний аналіз досліджуваного об’єкта

Дана система є багатофункціональною.

Функції першого рівня показані в таблиці 2.1.

Таблиця 2.1 – Функції системи першого рівня

Код	*Елемент*	*Функція*
1	Система антропометричних данних	Забезпечення введення антропометричних параметрів, аналізу і обробки цих параметрів, а так-же виведення результатів по введеним антро-пометріческім даними

Функції підсистем другого рівня представлені в таблиці 2.2.

Таблиця 2.2 - Функції підсистем другого рівня

Код	*Елемент*	*Функція*
1.1	Система формування вхідних параметрів	Введення антропометричних параметрів
1.2	Система аналізу антропометричних параметрів	Аналіз і обробка антропометричних параметрів
1.3	Модуль видачі результатів	Висновок результатів по введеним антропометричним даним

Функції третього рівня представлені в таблиці 2.3.

Таблиця 2.3 – Функції підсистем третього рівня

Код	*Елемент*	*Функція*
1.1.1	Підсистема виділення зовнішніх параметрів	Опис зовнішніх антропометричних параметрів людини
1.1.1	Підсистема виділення внутрішніх параметрів	Опис внутрішніх антропометричних параметрів людини
1.2.1	Модуль обробки вхідних параметрів	Обробка введених антропометричних параметрів
1.2.2	Модуль аналізу оброблених параметрів	Аналіз введених антропометричних параметрів

Параметри елементів першого рівня представлені в таблиці 2.4.

Таблиця 2.4 – Параметри систем першого рівня

Код	*Елемент*	*Параметри*
1	Система антропометричних параметрів	Модулі Антропометричні параметри Програма Формули

Параметри елементів другого рівня представлені в таблиці 2.5.

Таблиця 2.5 – Параметри підсистем другого рівня

Код	*Елемент*	*Параметри*
1.1	Система формування вхідних параметрів	Кількість вхідних параметрів Параметри щодо стану здоров'я Параметри щодо фізичної активності Параметри щодо будови організму
1.2	Система аналізу антропометричних параметрів	Модулі Середній час роботи модулів Формули
1.3	Модуль видачі результатів	Буквений висновок Оцінка якості життя Приблизний прогноз стану здоров'я людини

Параметри елементів третього рівня показані в таблиці 2.6

Таблиця 2.6. – Параметри підсистем третього рівня

Код	*Елемент*	*Параметри*

Продовження таблиці 2.6

1.1.1	Підсистема виділення зовнішніх параметрів	Ім'я, стать, вік, зріст, вага
1.1.2	Підсистема виділення внутрішніх параметрів	Тривалість сну, наявність шкідливих звичок, фізична активність
1.2.1	Модуль обробки вхідних параметрів	Програма, формули
1.2.2.	Модуль аналізу оброблених параметрів	Програма

Підсумкове і сумарна кількість функцій:

- функції першого рівня –1;
- функції другого рівня – 3;
- функції третього рівня –4;

Загальні характеристики системи:

Загальні характеристики заданої системи наведені в таблиці 2.7.

Таблиця 2.7. – Загальні характеристики системи

Функціональність	*Ранг*	*Фактори*	
		Системоруйнівні	*Системоутворюючі*
Багатофункціональна	Обслуговування	Неправильні вхідні дані	Правильні вхідні дані

2.3 Інформаційний опис системи

1. Елементи системи:

1) Система формування вхідних параметрів

2) Підсистема виділення зовнішніх параметрів

3) Підсистема виділення внутрішніх параметрів

4) Система аналізу антропометричних параметрів

5) Модуль обробки вхідних параметрів

6) Модуль аналізу оброблених параметрів

7) Модуль видачі результатів

2. Властивості системи

На таблиці 2.8 показані всі властивості системи.

Таблиця 2.8 – Властивості системи

№ п/п	*Найменування*	*Позначення*	*Кількість властивостей*	*Примітка*
1	Система формування вхідних параметрів	a_1	1	1 (1) забезпечення введення антропометричних параметрів (вхідних параметрів)
2	Підсистема виділення зовнішніх параметрів	a_2	2	1 (2) класифікація параметрів 1 (3) введення параметрів
3	Підсистема виділення внутрішніх параметрів	a_3	2	1 (4) класифікація параметрів 1 (5) введення параметрів
4	Система аналізу антропометричних параметрів	a_4	2	1 (6) забезпечення аналізу параметрів 1 (7) забезпечення обробки параметрів

Продовження таблиці 2.8

5	Модуль обробки вхідних параметрів	a_5	1	1 (8) обробка параметрів
6	Модуль аналізу оброблених параметрів	a_6	1	1(9) аналіз параметрів
7	Модуль видачі результатів	a_7	2	1 (10) забезпечення виведення оброблених і проаналізованих параметрів 1 (11) висновок оброблених і проаналізованих параметрів

1. Середньогеометричне число властивостей на один елемент:

$$A = \sqrt[7]{a_1 \cdot \dots \cdot a_8} = \sqrt[7]{1 \cdot 2 \cdot 2 \cdot 2 \cdot 1 \cdot 1 \cdot 2} = \sqrt[7]{16} \approx 1{,}485$$

2. Структура об'єкта мережева

Структура системи у вигляді графа представлена на рисунку 2.1

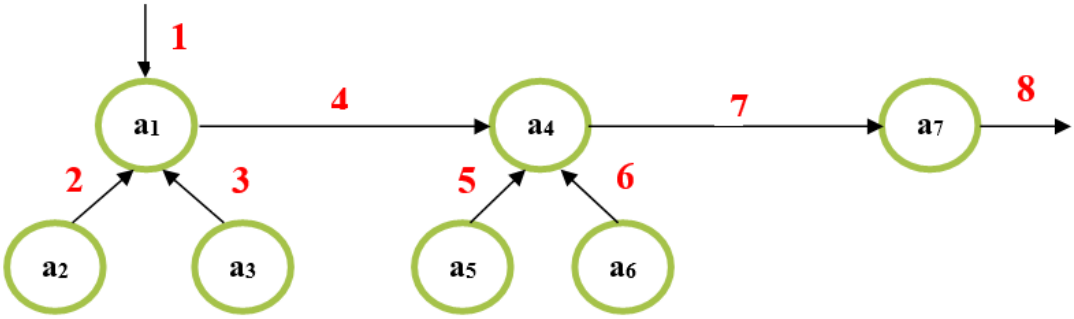

Рисунок 2.1 – Граф

5. Зв'язки системи між елементами:

А) Сполучні

2- підсистема виділення зовнішніх параметрів – система формування вхідних параметрів

3- підсистема виділення внутрішніх параметрів – система формування вхідних параметрів

4- система формування вхідних параметрів – система аналізу антропометричних параметрів

5- модуль обробки вхідних параметрів – система аналізу антропометричних параметрів

6- модуль аналізу оброблених параметрів – система аналізу антропометричних параметрів

7- система аналізу антропометричних параметрів – модуль видачі результатів

Б) Організуючі

1- зовнішнє середовище – система формування вхідних параметрів

8- модуль видачі результатів – зовнішнє середовище

6. Зв'язки системи

Зв'язки системи представлені на таблиці 2.9.

Таблиця 2.9 – Зв'язки системи

№ п/п	*Найменування*	*Кількість зв'язків*
1	Система формування вхідних параметрів	4
2	Підсистема виділення зовнішніх параметрів	1
3	Підсистема виділення внутрішніх параметрів	1
4	Система аналізу антропометричних параметрів	4
5	Модуль обробки вхідних параметрів	1
6	Модуль аналізу оброблених параметрів	1
7	Модуль видачі результатів	2

7. Середньогеометричне число зв'язків на один елемент:

$$Y = \sqrt[7]{4 \cdot 1 \cdot 1 \cdot 4 \cdot 1 \cdot 1 \cdot 2} = \sqrt[7]{32} \approx 1{,}641$$

2.4 Класифікаційний опис системи

Був проведен класифікаційний опис системи, який представлений у таблиці 2.10.

Таблиця 2.10 – Класифікаційний опис системи

№ п/п	*Ознака класифікації*	*Тип системи за ознакою*	*Визначення*
1	по зв'язку системи з навколишнім середовищем	відкрита	взаємодіє з навколишнім середовищем
2	за походженням	штучна	створена людиною
3	за об'єктивністю існування	реальна	складається з іскусственний (не живих) об'єктів; застосовується людиною в реального життя
4	за типом опису законів функціонування	система типу «білий ящик»	відомі всі закони функціонування
5	за способом управління системою	з комбінованим керуванням	системи підпорядковується блоку управління з системи, а також зовнішнього управління

Продовження таблиці 2.10

6	за дією	технічна	дана система - сукупність взаємопов'язаних фізичних елементів
7	по централізації	централізована система	не існує елемента, який є головним у функціонуванні системи
8	по однорідності структури	різнорідна	елементи системи не взаємозамінні
9	за типом складності	складна	має багато елементів і складні зв'язки
10	по мірній	многомірна	має багато входів і виходів
11	по організованості	добре організована	визначені всі елементи, зв'язки і залежності між елементами і цілями системи
12	по обумовленості дії	детермінована	введення об'єкта однозначно визначає його вихід
13	по лінійності	нелінійна	закон функціонування системи можна описати лінійним рівнянням

Продовження таблиці 2.10

14	по безперервності	безперервна	безперервний процес управління антропо-метричними даними

3 МОДЕЛЬНЕ УЯВЛЕННЯ ОБ’ЄКТА ТА ДОСЛІДЖЕННЯ ЙОГО ХАРАКТЕРИСТИК

3.1 Вхідні та вихідні параметри

Вхідні параметри і параметри на виході були складені на основі джерел [14, 15].

Вхідні параметри:

1. Зріст
2. Вага
3. Наявність шкідливих звичок
   3.1 Куріння
   3.2 Вживання алкоголю
   3.3 Вживання цукру
   3.4 Вживання кофеїну
4. Тривалість сну
5. Фізична активність

Параметри на виході:

1. Оцінка здоров'я людини
2. Надання рекомендацій
3. Попередження про наслідки

3.2 Модельне уявлення об’єкта дослідження

Головна мета системи аналізу антропометричних параметрів в цій роботі - це скорочення захворюваності і смертності людства. Для цього необхідно визначити основні параметри, за якими можна більш узагальнено визначити стан здоров'я людини. На основі робіт [16, 17] було встановлено, що для автоматизованої системи аналізу антропометричних параметрів, що розглядається в цій

роботі найбільш підходящими для аналізу будуть антропометричні дані, згадані вище в подглаве «3.1 Вхідні і вихідні величини».

При введенні цих параметрів в системі починають роботу модулі обробки, аналізу і видачі результатів. Обробивши параметри, модуль обробки передає інформацію модулю аналізу, який аналізує її і передає інформацію модулю видачі результатів. Весь процес представлений на рисунку 3.1.

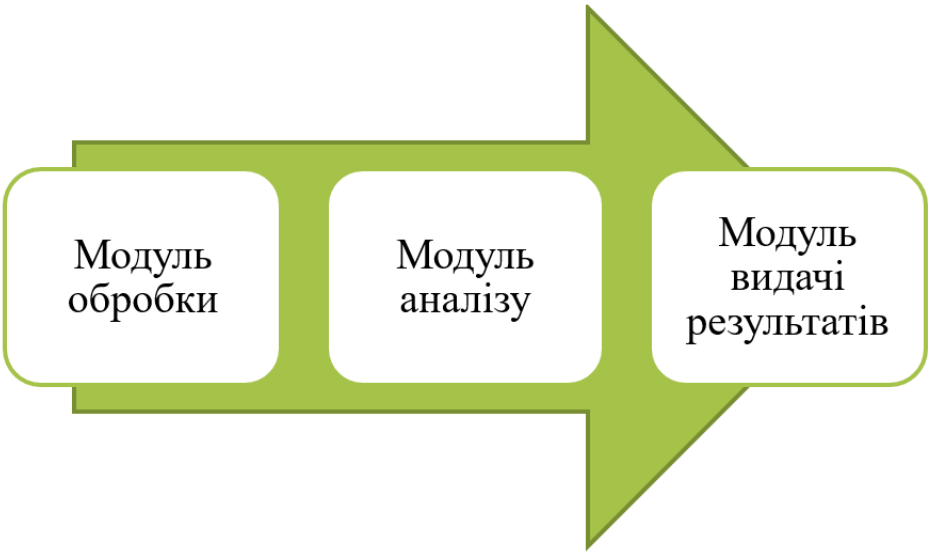

Рисунок 3.1 – Модель роботи системи

3.3 Складання правил

3.3.1 Зріст та вага

Індекс маси тіла розраховується за формулою 4.1.

$$I = \frac{m}{h^2} \tag{4.1}$$

де: m — маса тіла у кілограмах , h — зріст у метрах.

- I =[10, 18,5]

Оцінка: дефіцит і недостатня маса тіла

Наслідки: надмірна худорба може привести до розладів нервової системи, різних захворюваннь шлунково-кишкового тракту, ендокринним і гормональним порушенням, зниженню імунітету; шкіра почне втрачати пружність, з'являться зморшки і розтяжки; порушиться робота термогенеза - тобто здатності організму виробляти тепло для підтримки постійної температури тіла і забезпечення роботи всіх його систем.

Рекомендації:

1) добре харчуйтеся і намагайтеся робити це регулярно, 4-5 разів на добу
2) відмовтеся від фаст-фуду і краще віддайте перевагу йогуртам і сухофруктам, овочам і фруктам; включите в свій раціон продукти бога-ті білками і вуглеводами; щоб набрати вагу, необхідно щодоби споживати 2,5 м білка на 1 кг. ваги
3) постарайтеся висипатися: здоровий повноцінний сон сприяє ви-розробці соматотропіну - гормону, що впливає на приріст м'язової маси
4) займайтеся спортом
5) ведіть "щоденник харчування і фізичних навантажень»
6) відмовтеся від куріння і вживання алкоголю: вони погіршують апетит, знижують стресостійкість, що негативно впливає на обмін речовин; з цієї ж причини варто скоротити споживання кави і міцного чаю
7) постарайтеся позбутися від стресів: нервове перенапруження «спалює калорії», а це заважає набрати вагу

- I=[18,5,25]

Оцінка: норма ваги

Рекомендації: у вас мабуть чудово виходить тримати свій організм в тонусі! Але не забувайте про правильне, збалансоване харчування і про заняття спортом.

- I=[25,30]

Оцінка: надлишкова маса тіла

Наслідки: серцево-судинні захворювання (головним чином, хвороби серця); діабет; порушення опорно-рухової системи.

Рекомендації:

1) слід врівноважити кількість споживаної їжі і рівень фізичних навантажень
2) обмежити калорійність свого раціону за рахунок зниження кількості по-требляют жирів і цукрів
3) збільшити споживання фруктів і овочів, а також зернобобових, цільних злаків і горіхів
4) вести регулярну фізичну активність
5) вести «щоденник харчування і фізичних навантажень»

- I=[30,60]

Оцінка: ожиріння

Наслідки: серцево-судинні захворювання (головним чином, хвороби серця та інсульт); діабет; порушення опорно-рухової системи (особливо остеоартрит); деякі онкологічні захворювання.

Рекомендації:

1) слід врівноважити кількість споживаної їжі і рівень фізичних навантажень
2) обмежити калорійність свого раціону за рахунок зниження кількості по-требляют жирів і цукрів

3) збільшити споживання фруктів і овочів, а також зернобобових, цільних злаків і горіхів

4) вести регулярну фізичну активність (60 хвилин в день для дітей і 150 хвилин в тиждень для дорослих)

5) вести «щоденник харчування і фізичних навантажень»

6) звернутися до психолога, якщо ви відчуваєте що їжа є вашим спасенням від стресів, нудьги, відчуття небезпеки, тривоги або самотності.

3.3.2 Тривалість сну

- 1-5 годин

Оцінка: недосипання

Наслідки:

1) зниження імунітету
2) зниження працездатності, концентрації уваги, пам'яті
3) серцево-судинні захворювання
4) головні болі
5) ожиріння (організм, немов захищаючись, намагається заповнити недолік енергії зайвими калоріями)
6) у чоловіків внаслідок недосипу знижується рівень тестостерону на 30% (починає рости живіт навіть у худорлявих чоловіків, з'являється ризик воспалення передміхурової залози)
7) підвищується рівень гормону стресу - кортизолу
8) може розвинутися депресія, безсоння

Рекомендації: Вам варто збільшити години сну, хоча б до 6 годин, а краще приділяти 8 годин для сну, так як недосип може привести до наслідків, описаним вище.

- 6-8 годин

Оцінка: норма сну

Рекомендації: вчені вважають, що це нормальне кількість годин сну і це не зашкодить Вашому організму, так що Ви - молодець!

- 9 і більше

Оцінка: пересипання

Наслідки:

1) цукровий діабет (ризик захворіти діабетом підвищувався на 50% у людей, що спали більше дев'яти годин на добу)
2) проблеми з серцево-судинною системою (ризик ішемічної хвороби підвищується на 38% у людей, які сплять від 9 до 11 годин на добу)
3) ожиріння
4) головні болі
5) дратівливість, поганий настрій
6) депресія

Рекомендації: Ви спите занадто багато! Вам потрібно встановити режим сну і приділяти йому не більше 8-ми годин, інакше пересип може привести до наслідків, описаним вище.

3.3.3 Фізична активність

За п'ятибальною шкалою (0-мінімум, 5-максимум) оцінити: як часто проходять заняття спортом -x; фізичний стан-y.

- При x[0,2] і y[0 ,2]

Оцінка: малорухливий спосіб життя, який призводить до нестабільного фізичному стану тіла.

Наслідки: старіння дихальної та серцево-судинної системи, ослаблення скелета, хвороби суглобів, млявість, слабкість, апатія і ожиріння.

Рекомендації: Вам рекомендується приділити підвищену увагу заняттю спортом або, для початку, намагайтеся більше ходити пішки і прогулюватися перед сном, що буде також корисно і для хорошого міцного сну.

- При x=3 і y=3

Оцінка: співвідношення кількості фізичних навантажень і загального стану тіла у нормі.

Рекомендації: можете спробувати збільшити фізичні навантаження, це може посприяти поліпшенню Вашого стану.

- При x[4,5] і y[4,5]

Оцінка: висока фізична активність, що приводить до тонусу і хорошого показником фізичного стану тіла.

Рекомендації: продовжуйте у тому ж дусі!

- При x=3 і y[4,5]

Оцінка: нормальна фізична активність, організм знаходиться в тонусі.

Рекомендації: можете спробувати збільшити свої фізичні навантаження, але Ви і так молодець!

- При x[4,5] і y=3

Оцінка: загальний стан Вашого тіла не відповідає фізичним навантаженням, які Ви виконуєте.

Наслідки: погіршення сну, втрата ваги, травма м'язів.

Рекомендації: постарайтеся знизити фізичну активність і поспостерігати за станом тіла. Якщо воно не покращиться - зверніться до лікаря. Якщо покращиться, значить Ви перевищували свою норму заняття спортом і Вам варто ретельніше добирати собі програму для фіз.нагрузок.

- При x[0,2] і y[4,5]

Оцінка: низька фізична активність, організм зберігає хороший фізичний стан.

Наслідки: старіння дихальної та серцево-судинної системи, ослаблення скелета, хвороби суглобів.

Рекомендації: це добре, що Ваше тіло знаходиться в тонусі при мінімальному фізичному навантаженні, але Ви повинні розуміти, що при низькій фізичної активності або зовсім при її відсутності організм не завжди буде так добре себе почувати. Спробуйте активніше вести свій спосіб життя!

- При x[4,5] і y [0,2]

Оцінка: загальний стан Вашого тіла не відповідає фізичним навантаженням, які Ви виконуєте.

Наслідки: погіршення сну, втрата ваги, травма м'язів, часті виникненням гострих респіраторних вірусних захворювань (процес одужання триватиме довше звичайного).

Рекомендації: Вам необхідно зменшити фізичні навантаження, не займайтеся через силу і зверніться за допомогою до тренера, щоб той грамотно склав Вам режим тренувань і допоміг з вибором певних навантажень.

- При x[0,2] і y=3

Оцінка: низька фізична активність і нормальний фізичний стан тіла.

Наслідки: старіння дихальної та серцево-судинної системи, ослаблення скелета, хвороби суглобів.

Рекомендації: Вам не завадило б підвищити свої фіз.нагрузкі, так як при низькій фізичної активності або зовсім при її відсутності організм не завжди буде нормально себе почувати. Спробуйте активніше вести свій спосіб життя!

- При x=3 і y[0,2]

Оцінка: нормальна фізична активність і низька фізичний стан тіла.

Наслідки: млявість, слабкість.

Рекомендації: Вам слід звернутися за консультацією до лікаря або до тренера, який би зміг грамотно скласти режим тренувань і допоміг з вибором певних завантажень.

3.3.4 Шкідливі звички

За п'ятибальною шкалою, де 0-мінімум, 5-максимум, оцінити як часто відбувається вживання ніжеперечсіленних шкідливих звичок.

Куріння:

- 0

Оцінка: відсутність нікотинової залежності.

Рекомендації: Ви відмінно дотримуєтеся приказки «Курити - здоров'ю шкодити!», Так тримати!

- [1,2]

Оцінка: помірне споживання сигарет.

Наслідки:

1) рак (найбільш негативними наслідками пристрасті до сигарет є онкологічні захворювання бронхів, легенів, трахеї, гортані, стравоходу, сечового міхура і підшлункової залози. Крім того, страждають нирки, органи репродуктивної та кровотворної систем)
2) захворювання серцево-судинної системи (це такі наслідки куріння, як ішемічна хвороба серця, хвороба Бюргера, порушення в периферичних судинах, інсульти, тромбози та ін)
3) патології органів травлення (куріння шкодить і здоров'ю шлунково-кишкового тракту, викликаючи утворення поліпів товстого кишечника, виразки шлунка і дванадцятипалої кишки, гастрит, гастродуоденіт і ін.)
4) хвороби дихальної системи (куріння сигарет провокує розвиток або ускладнює перебіг бронхіальної астми, хронічного риніту, туберкул через, хронічної обструктивної хвороби легень і бронхіту, а також збільчивает частоту захворюваності на ГРЗ та грип)
5) захворювання порожнини рота (наслідком куріння сигарет може стати не толь-ко пожовтіння емалі, але і такі серйозні патології, як

некротичний виразковий гінгівіт, пародонтит, онкологічні поразки слизових оболонок)

6) порушення опорно-рухового апарату (куріння сигарет надає негативний вплив і на скелет людини. Воно згубно позначається на стані сухожиль і зв'язок, а також м'язової тканини. Під впливом куріння в організмі погіршується засвоєння кальцію, розвивається остеопороз, зростає частота переломів і ризик формування ревматоїдного артриту)

7) хвороби очей (небезпека куріння полягає і в провокуванні таких патологій, як макулярна дистрофія (ураження сітківки), ністагм (аноформальні руху очних яблук), тютюнова амбліопія (втрата зору), діабетична ретинопатія (ураження судин сітківки очей при цукровому діабеті) , катаракта та ін.)

8) захворювання репродуктивної системи (куріння шкідливе і для статевих органів. Найбільш частими наслідками у жінок є менструальні дисфункції, зниження фертильності, ановуляторні цикли, рання менопауза. Під впливом куріння здоров'я чоловіків страждає не менше. У них відзначається зниження фертильності, еректильна дисфункція, зменшення кількості сперматозоїдів в спермі, погіршення їх якості і рухливості)

9) цукровий діабет II типу, депресія, розсіяний склероз, порушення слуху та інші недуги.

Рекомендації: не дивлячись на те, що Ви не запеклий курець, все ж краще відмовити від цієї звички, щоб запобігти перераховані вище проблеми зі здоров'ям. Задумайтесь про своє здоров'я!

- [3,4]

Оцінка: часте споживання сигарет.

Наслідки:

1) рак (найбільш негативними наслідками пристрасті до сигарет є онкологічні захворювання бронхів, легенів, трахеї, гортані, стравоходу, сечового міхура і підшлункової залози. Крім того, страждають нирки, органи репродуктивної та кровотворної систем)
2) захворювання серцево-судинної системи (це такі наслідки куріння, як ішемічна хвороба серця, хвороба Бюргера, порушення в периферичних судинах, інсульти, тромбози та ін)
3) патології органів травлення (куріння шкодить і здоров'ю шлунково-кишкового тракту, викликаючи утворення поліпів товстого кишечника, виразки шлунка і дванадцятипалої кишки, гастрит, гастродуоденіт і ін.)
4) хвороби дихальної системи (куріння сигарет провокує розвиток або ускладнює перебіг бронхіальної астми, хронічного риніту, туберкул через, хронічної обструктивної хвороби легень і бронхіту, а також збільчивает частоту захворюваності на ГРЗ та грип)
5) захворювання порожнини рота (наслідком куріння сигарет може стати не толь-ко пожовтіння емалі, але і такі серйозні патології, як некротичний виразковий гінгівіт, пародонтит, онкологічні поразки слизових оболонок)
6) порушення опорно-рухового апарату (куріння сигарет надає негативний вплив і на скелет людини. Воно згубно позначається на стані сухожиль і зв'язок, а також м'язової тканини. Під впливом куріння в організмі погіршується засвоєння кальцію, розвивається остеопороз, зростає частота переломів і ризик формування ревматоїдного артриту)
7) хвороби очей (небезпека куріння полягає і в провокуванні таких патологій, як макулярна дистрофія (ураження сітківки), ністагм (аноформальні руху очних яблук), тютюнова амбліопія (втрата зору), діабетична ретинопатія (ураження судин сітківки очей при цукровому діабеті) , катаракта та ін.)
8) захворювання репродуктивної системи (куріння шкідливе і для статевих органів. Найбільш частими наслідками у жінок є менструальні

дисфункції, зниження фертильності, ановуляторні цикли, рання менопауза. Під впливом куріння здоров'я чоловіків страждає не менше. У них відзначається зниження фертильності, еректильна дисфункція, зменшення кількості сперматозоїдів в спермі, погіршення їх якості і рухливості)

9) цукровий діабет II типу, депресія, розсіяний склероз, порушення слуху та інші недуги.

Рекомендації: не дивлячись на те, що у Вас ще немає залежності від сигарет, все ж краще відмовити від цієї звички, щоб запобігти перераховані вище проблеми зі здоров'ям. Задумайтесь про своє здоров'я!

- 5

Оцінка: залежність від сигарет.

Наслідки:

1) рак (найбільш негативними наслідками пристрасті до сигарет є онкологічні захворювання бронхів, легенів, трахеї, гортані, стравоходу, сечового міхура і підшлункової залози. Крім того, страждають нирки, органи репродуктивної та кровотворної систем)

2) захворювання серцево-судинної системи (це такі наслідки куріння, як ішемічна хвороба серця, хвороба Бюргера, порушення в периферичних судинах, інсульти, тромбози та ін)

3) патології органів травлення (куріння шкодить і здоров'ю шлунково-кишкового тракту, викликаючи утворення поліпів товстого кишечника, виразки шлунка і дванадцятипалої кишки, гастрит, гастродуоденіт і ін.)

4) хвороби дихальної системи (куріння сигарет провокує розвиток або ускладнює перебіг бронхіальної астми, хронічного риніту, туберкул через, хронічної обструктивної хвороби легень і бронхіту, а також збільчивает частоту захворюваності на ГРЗ та грип)

5) захворювання порожнини рота (наслідком куріння сигарет може стати не тільки пожовтіння емалі, але і такі серйозні патології, як некротичний виразковий гінгівіт, пародонтит, онкологічні поразки слизових оболонок)

6) порушення опорно-рухового апарату (куріння сигарет надає негативний вплив і на скелет людини. Воно згубно позначається на стані сухожиль і зв'язок, а також м'язової тканини. Під впливом куріння в організмі погіршується засвоєння кальцію, розвивається остеопороз, зростає частота переломів і ризик формування ревматоїдного артриту)

7) хвороби очей (небезпека куріння полягає і в провокуванні таких патологій, як макулярна дистрофія (ураження сітківки), ністагм (аноформальні руху очних яблук), тютюнова амбліопія (втрата зору), діабетична ретинопатія (ураження судин сітківки очей при цукровому діабеті) , катаракта та ін.)

8) захворювання репродуктивної системи (куріння шкідливе і для статевих органів. Найбільш частими наслідками у жінок є менструальні дисфункції, зниження фертильності, ановуляторні цикли, рання менопауза. Під впливом куріння здоров'я чоловіків страждає не менше. У них відзначається зниження фертильності, еректильна дисфункція, зменшення кількості сперматозоїдів в спермі, погіршення їх якості і рухливості)

9) цукровий діабет II типу, депресія, розсіяний склероз, порушення слуху та інші недуги.

Рекомендації як позбутися залежності сигарет:

1) Призначте дату, що має для вас якийсь особливий сенс, якщо ця дата близька. Це може бути ваш день народження, день народження подруги (друга). Новий рік або якась річниця. Якщо ви курите через стрес, викликаного навчанням, кидайте цю звичку під час канікул. Не

призначайте дату в віддаленому майбутньому, ви можете втратити душевний запал.

2) Домовтеся з кращим другом (подругою) або дружиною (чоловіком) бро-сить курити разом, щоб ви могли підтримувати один одного.

3) Скажіть усім вашим знайомим, що ви кидаєте палити. Вони будуть намагатися вас підтримати.

4) Визначте коло людей (що підтримують вас в прагненні кинути курить), яким ви можете зателефонувати в будь-який час, коли вам дуже захочеться покурити.

5) Спробуйте замінити куріння іншими заняттями - фізичними упражненіями, новим захопленням, жувальною гумкою або низькокалорійними закусками. Уникайте є висококалорійну їжу: можна набрати зайву вагу.

6) Найкраще кинути палити відразу і повністю. Поступове позбавлення від звички палити дає гірші результати. Однак ті, хто придбав пристрасть до нікотину, можуть виходити з куріння поступово (або застосовувати нікотинову жуйку) з тим, щоб уникнути синдрому втрати. Якщо ви збираєтеся припиняти курити поступово, заздалегідь розробіть схему і твердо їй випливайте.

7) Чи не запалюйте сигарету, поки не пройде 5 хв з моменту виникнення у вас потреби покурити. Протягом цих 5 хв спробуйте змінити свій емоційний настрій або зайнятися чимось іншим. Зателефонуйте комунебудь з вашої «групи підтримки».

8) Зробіть куріння настільки незручним, наскільки це можливо. Завжди купуйте тільки одну пачку сигарет і тільки після того, як закінчилася попередня. Ніколи не носіть сигарети при собі - ні вдома, ні на робо-ті. Не носіть при собі сірники або запальнички.

9) Складіть список речей, які можна було б купити на зекономлені на палінні гроші. Переведіть вартість кожної з них в дні не куріння.

Алкоголь:

- 0

Оцінка: відсутність залежності від алкоголю.

Рекомендації: здорово, що ви не вживаєте алкоголь, Ви - молодець, продовжуйте в тому ж дусі!

- [1,2]

Оцінка: помірне споживання алкоголю.

Наслідки:

1) Серцево-судинні (підвищення артеріального тиску, інфаркт міокарда та інсульт)
2) Хвороби печінки (стеатоз, алкогольний гепатит і цироз печінки)
3) Виразкова хвороба шлунка та дванадцятипалої кишки.
4) Гострий і хронічний панкреатит (ураження підшлункової залози).
5) Рак губи і горла (медичні дослідження доводять: у людей, які зловживають алкоголем, може розвиватися рак губи і горла. Ризик значно збільшується, якщо той, хто зловживає алкоголем, ще й курить)
6) Психологічні проблеми і розвиток депресії

Рекомендації: Вам слід кинути пити, якщо Ви не хочете стати залежним від алкоголю і не отримати перераховані вище наслідки! Алкоголь - не друг, йому не повинно бути місця в житті здорового і люблячої себе людини!

- [3,4]

Оцінка: часте споживання алкоголю.

Наслідки:

1) Серцево-судинні (підвищення артеріального тиску, інфаркт міокарда та інсульт)

2) Хвороби печінки (стеатоз, алкогольний гепатит і цироз печінки)
3) Виразкова хвороба шлунка та дванадцятипалої кишки.
4) Гострий і хронічний панкреатит (ураження підшлункової залози).
5) Рак губи і горла (медичні дослідження доводять: у людей, які зловживають алкоголем, може розвиватися рак губи і горла. Ризик значно збільшується, якщо той, хто зловживає алкоголем, ще й курить)
6) Психологічні проблеми і розвиток депресії

Рекомендації: Вам слід кинути пити, якщо Ви не хочете стати залежним від алкоголю і не отримати перераховані вище наслідки! Алкоголь - не друг, йому не повинно бути місця в житті здорового і люблячої себе людини!

- 5

Оцінка: залежність від алкоголю.

Наслідки:

1) Серцево-судинні (підвищення артеріального тиску, інфаркт міокарда та інсульт)
2) Хвороби печінки (стеатоз, алкогольний гепатит і цироз печінки)
3) Виразкова хвороба шлунка та дванадцятипалої кишки.
4) Гострий і хронічний панкреатит (ураження підшлункової залози).
5) Рак губи і горла (медичні дослідження доводять: у людей, які зловживають алкоголем, може розвиватися рак губи і горла. Ризик значно збільшується, якщо той, хто зловживає алкоголем, ще й курить)
6) Психологічні проблеми і розвиток депресії

Рекомендації як позбутися залежності алкоголю:

1) Не піддавайтеся самообману. Відмова від спиртного - це праця, одного рішення змінити життя - мало, треба добре усвідомлювати, що на перших порах буде важко, але результат вартий того!
2) Відмовтеся від інших залежностей

3) Навчіться розслаблятися без алкоголю
4) Займіться спортом
5) Правильно і регулярно харчуйтеся
6) Позбавтеся від нудьги
7) Ізолюйтеся від спокус

Цукор:

- 0

Оцінка: відсутність залежності від цукру.

Рекомендації: здорово, що у Вас немає залежності від цукру! З Вас можна брати приклад!

- [1,2]

Оцінка: помірне споживання цукру.

Наслідки:

1) Цукор викликає відкладення жиру
2) Цукор створює почуття помилкового голоду
3) Цукор сприяє старінню
4) Цукор викликає звикання
5) Цукор позбавляє організм вітамінів групи В
6) Цукор впливає на серце
7) Цукор виснажує енергетичний запас
8) Цукор є стимулятором
9) Цукор вимиває кальцій з організму

Рекомендації: Ви не так вже й багато споживаєте цукор, молодець! Але краще не забувати, чим загрожує часте споживання цукру, щоб не хотілося їсти його ще більше!

- [3,4]

Оцінка: часте споживання цукру.

Наслідки:

1) Цукор викликає відкладення жиру
2) Цукор створює почуття помилкового голоду
3) Цукор сприяє старінню
4) Цукор викликає звикання
5) Цукор позбавляє організм вітамінів групи В
6) Цукор впливає на серце
7) Цукор виснажує енергетичний запас
8) Цукор є стимулятором
9) Цукор вимиває кальцій з організму

Рекомендації: так Ви ласун! Вам краще б знизити кількість споживання цукру, щоб не виникли перераховані вище проблеми.

- 5

Оцінка: залежність від сахару.

Наслідки:

1) Цукор викликає відкладення жиру
2) Цукор створює почуття помилкового голоду
3) Цукор сприяє старінню
4) Цукор викликає звикання
5) Цукор позбавляє організм вітамінів групи В
6) Цукор впливає на серце
7) Цукор виснажує енергетичний запас
8) Цукор є стимулятором
9) Цукор вимиває кальцій з організму

Рекомендації: так Ви ласун! Якщо Ви дбаєте про своє здоров'я і не хочете виникнення вищеперелічених проблем, то краще знизити споживання цукру! Можете замінити його на більш корисні продукти: мед, кленовий сироп, сухофрукти.

Кофеїн:

- 0

Оцінка: відсутність залежності від кофеїну.

Рекомендації: вау, це круто, що Ви не вживаєте кофеїн, так тримати!

- [1,2]

Оцінка: помірне споживання кофеїну.

Наслідки:

1) Прискорене серцебиття
2) Нервове тремтіння
3) Безсоння
4) Підвищений кров'яний тиск

Рекомендації: добре, що Ви мало споживаєте кофеїну! Але краще знати, які можуть бути наслідки від його частого вживання, щоб при питанні «А може ще по чашці кави?», Ви точно говорили «Ні!».

- [3,4]

Оцінка: часте споживання кофеїну.

Наслідки:

1) Прискорене серцебиття
2) Печія
3) Нервове тремтіння

4) Безсоння
5) Підвищений кров'яний тиск

Рекомендації: ага, мабуть Ви любитель чашечки кави вранці, а може навіть і не однієї! Постарайтеся зменшити споживання кофеїну, щоб він не викликав у Вас перераховані вище проблеми зі здоров'ям.

- 5

Оцінка: залежність від кофеїну.

Наслідки:

1) Прискорене серцебиття
2) Печія
3) Нервове тремтіння
4) Діарея
5) Безсоння
6) Підвищений кров'яний тиск
7) Пошкодження печінки

Рекомендації: краще позбутися такої залежності, але будьте обережні, в клініці Мейо попереджають, що різка відмова від кави може викликати симптоми відміни, такі як головний біль, втому і дратівливість, так що краще робити це поступово.

3.4 Технології IoT

Інтернет речей (англ. Internet of Things, IoT)— концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами в автоматичному режимі, за допомогою використання стандартних протоколів зв'язку. Окрім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між

собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Основною концепцією IP є можливість підключення всіляких об'єктів, які людина може використовувати в повсякденному житті, наприклад, холодильник, кондиціонер, автомобіль, велосипед і навіть кросівки. Всі ці об'єкти повинні бути оснащені вбудованими давачами або сенсорами, які мають можливість обробляти інформацію, що надходить з навколишнього середовища, обмінюватися нею і виконувати різні дії в залежності від отриманої інформації. Прикладом впровадження такої концепції є система «розумний будинок» або «розумна ферма». Ця система аналізує дані навколишнього середовища і в залежності від показників регулює температуру в приміщенні. У зимовий період регулюються інтенсивність опалення, а в разі спекотної погоди будинок має механізми відкривання і закривання вікон, завдяки чому провітрюється будинок, і все це відбувається без втручання людини.

3.4.1 Медичне обладання

Інтернет медичних речей (IoMT) являє собою інфраструктуру розумних пристроїв, ПО, систем охорони здоров'я та окремих смарт-послуг. IoMT удосконалює і розвиває галузь охорони здоров'я, допомагає надавати допомогу віддалено, збирати більше інформації про клієнта.

Смарт-пристрої (наприклад, розумні гаджети, датчики, вимірювачі серцевого ритму) збирають і обробляють дані. Вони допомагають медичному персоналу оперативно формувати персональну статистику пацієнтів, їхні медичні карти. При цьому лікарі можуть застосовувати інформацію з підключених систем, щоб ефективніше лікувати пацієнтів, надавати їм своєчасний і відповідний догляд, а також попереджати загострення хронічних захворювань.

3.4.2 Розумні ваги

Розумні ваги використовують біоімпедансний аналіз тіла. Ця процедура зі сложнопроізносімих назвою дозволяє визначити процентне співвідношення різних тканин в організмі за допомогою електрики.

Пропускаючи по тілу слабкий заряд змінного струму порівняно високої частоти (до 100 кГц), можна потім реєструвати, як цей струм проходить через весь організм. В результаті таких вимірювань виходять числа, які самі по собі ще нічого не говорять про склад тіла - зібрану інформацію необхідно ще належним чином проаналізувати.

Заздалегідь знаючи значення імпедансу різних тканин тіла, які у всіх людей однакові, ми можемо оцінити і процентне співвідношення цих тканин у однієї конкретної людини. Для цього існують спеціальні формули і програми, закладені в пам'ять "розумних" ваг або пов'язаних з ними програм. Якщо співвіднести отриману інформацію з вашими особистими даними (вік, вага), смарт-ваги зможуть надати користувачеві заповітні дані про те, з чого складається його тіло.

Як правило, "розумні" ваги поділяють весь ваш вага на 5 складових: м'язи, жир, кістки, рідина і все інше. Деякі ваги показують також значення активної клітинної маси, що включає в себе м'язи, внутрішні органи і нервові клітини. У процесі схуднення важливо, щоб активна клітинна маса не зменшувалася, так як саме вона відповідальна за спалювання жирів у вашому організмі.

3.4.3 Розумний годинник

Розумний годинник — комп'ютеризований наручний пристрій, який окрім вимірювання часу виконує додаткові функції: відтворення музики, прийом сповіщень та дзвінків з мобільного телефону, відстеження маршрутів, збір інформації з вбудованих та зовнішніх датчиків (наприклад, акселерометру, термометру) тощо.

Пристрої для загальних потреб можуть виконувати функції камери, акселерометру, термометру, хронографу, калькулятору, GPS навігатору, плеєру,

мати доступ до Інтернету. Спортивні годинники, окрім названих функцій, мають додаткові, призначені для ведення активного способу життя. Спортивні годинники сумісні з моніторами серцевого ритму, датчиками обертання педалей велосипеда. Спеціальні програми дозволяють відстежувати та записувати фізичну активність користувача.

3.5 Обробка даних

Apache Kafka - розподілене програмний брокер повідомлень, проект з відкритим вихідним кодом, що розробляється в рамках фонду Apache. Написаний на мовах програмування Java і Scala.

Спроектовано як розподілена, горизонтально масштабована система, що забезпечує нарощування пропускної здатності як при зростанні числа і навантаження з боку джерел, так і кількості систем-передплатників. Передплатники можуть бути об'єднані в групи. Підтримується можливість тимчасового зберігання даних для подальшої пакетної обробки. Однією з особливостей реалізації інструменту є застосування техніки, схожої з журналами транзакцій, використовуваними в системах управління базами даних.

Apache Spark Streaming - компонент Spark для обробки потокових даних. Модуль Spark Streaming побудований із застосуванням «мікропакетной» архітектури (micro-batch architecture), коли потік даних інтерпретується як безперервна послідовність маленьких пакетів даних. Spark Streaming приймає дані з різних джерел і об'єднує їх в невеликі пакети. Нові пакети створюються через регулярні інтервали часу. На початку кожного інтервалу часу створюється новий пакет, і будь-які дані, що надійшли протягом цього інтервалу, включаються в пакет. В кінці інтервалу збільшення пакета припиняється.

3.6 Методи оптимізації

До критеріїв оптимізації слід віднести:

• правильний підбір антропометричних параметрів;

• узагальнений аналіз параметрів;

• коректна робота модулів системи;

• введення нових параметрів, що вимагають вимір людини.

Дані критерії дозволять більш точно і якісно спрогнозувати стан здоров'я людини і оцінити його якість життя.

4 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вибір платформи для розробки програмного продукту

Операційна система – це системне програмне забезпечення, яке управляє комп'ютерними апаратними та програмними ресурсами і надає загальні служби для комп'ютерних програм. Усі комп'ютерні програми, за винятком прошивки, вимагають роботи операційної системи.

Вибір платформи – фундаментальна проблема для будь-якого нового проекту. Android, iOS, Windows або що-небудь інше [38]?

На рисунку 4.1 показана діаграма використання різних операційних систем в Україні. Дані актуальні на початок 2018 року.

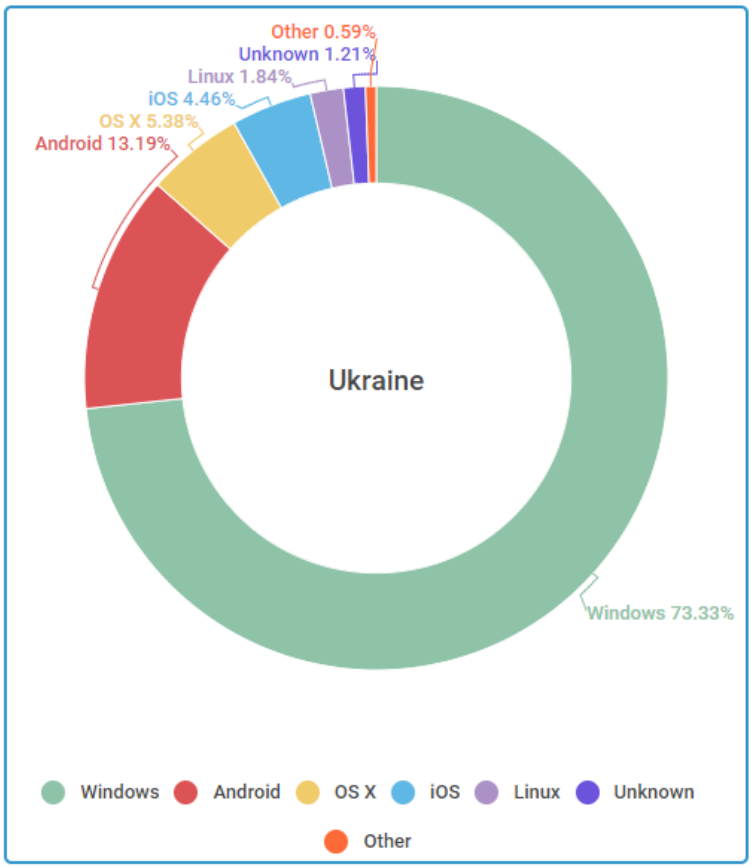

Рисунок 4.1 – Статистична інформація про користувачів операційних систем в Україні

У секторі мобільних пристроїв за даними третього кварталу 2017 року Android від Google домінує з 87,5%, а темпи зростання - 10,3% в рік, за яким слід iOS Apple з 12,1% і в рік зниження ринку Частка в 5,2 відсотки, тоді як інші операційні системи складають всього 0,3 відсотка [38].

В Україні лідируючої по використанню операційною системою є Windows - 73.33% всіх пристроїв, від настільних до кишенькових. І так як мобільні технології беруть своє, в Україні теж взяв 13,19%. Операційна система виробництва компанії Apple OS X займає третє місце - 5.38%, і навіть якщо до нього додати показник iOS - 4,46% - це не дає йому обігнати за популярністю в Україні Android.

На рисунку 4.2 показана діаграма використання різних операційних систем у всьому світі. Дані актуальні на початок 2018 року.

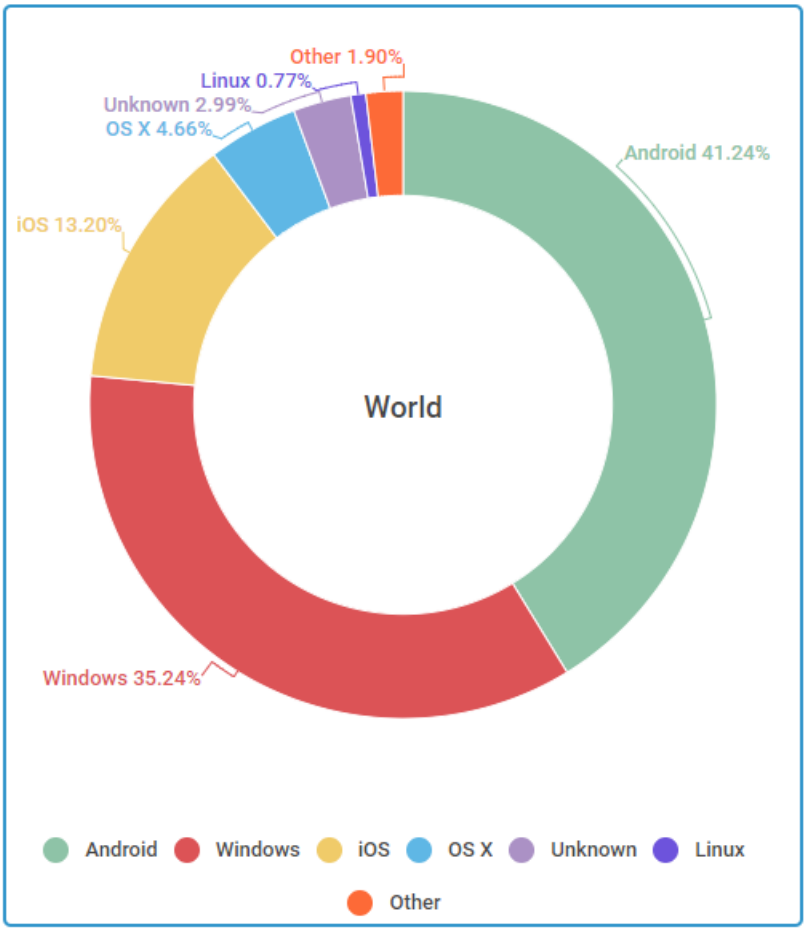

Рисунок 4.2 – Статистична інформація про користувачів операційних систем в усьому світі

З огляду на сукупну статистику настільних і мобільних платформ, лідирує OS Android - 41.24% всіх пристроїв. Другий і третій відповідно Windows - 35,24% і iOS - 13,2%.

Виходячи з статистичної інформації, можна зробити висновок про те, що краще розробляти багатоплатформений додаток, так як існує великий попит як на Windows, так і на Android та iOS. Багатоплатформений додаток дозволить бути використаним користувачами будь-яких операційних систем, що є відмінним рішенням для оптимального користування додатком.

4.2 Вибір архітектури ПЗ

4.2.1 Клієнт-серверна архітектура

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними [35]. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів [35].

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів — клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін [35].

4.2.2 MVC

Модель–вигляд–контролер — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними,

а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [37].

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності [37]. На рисунку 4.3 зображена діаграма взаємодії між компонентами шаблону.

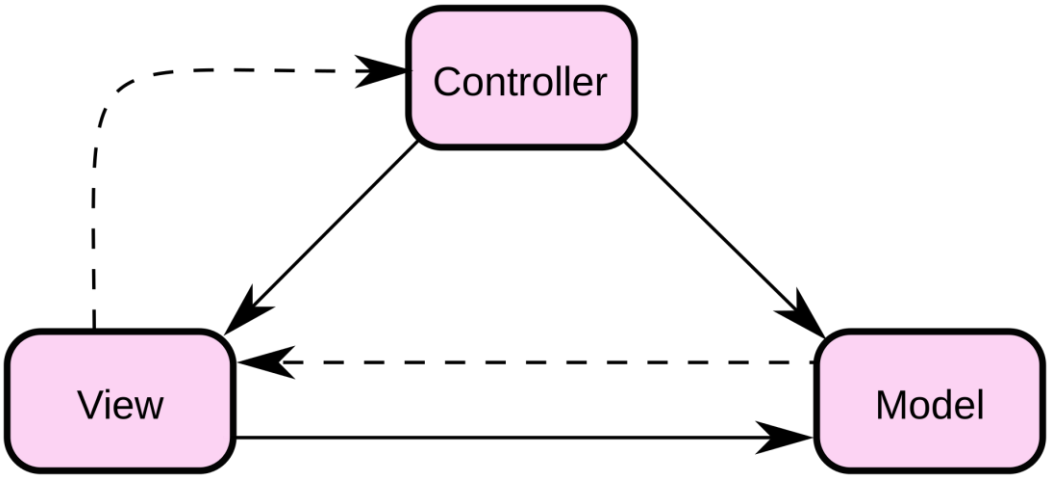

Рисунок 4.3 – Діаграма взаємодії між компонентами шаблону

У рамках архітектурного шаблону MVC програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель [37].

- Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
- Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть

співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

- Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

4.2.3 Інверсія контролю

Інверсія керування — це принцип побудови програми, при якому її частини отримують потік керування (викликаються) із загальної спільновикористовуваної бібліотеки. Це ніби звичайне процедурне програмування вивернуте навиворіт (inversed). Також це називають «голлівудським принципом»: «Не дзвоніть нам, ми подзвонимо вам» [34].

Однією з реалізацій IoC є впровадження залежностей (англ. Dependency Injection), що використовується в багатьох фреймворках, вони називаються IoC контейнери. Використовуються в таких об'єктно-орієнтованих мовах програмування, як Smalltalk, C++, Java, PHP або мови платформи .NET.

Наприклад, у випадку традиційного програмування, головна функція програми може викликати функцію із бібліотеки, щоб відобразити список доступних команд, і запросити користувача вибрати одну з них. Бібліотека поверне вибрану опцію як результат виклику функції. Цей стиль використовувався у текстових інтерфейсах. Наприклад, поштовий клієнт може показати екран з командами для завантаження нових листів, відповіді на поточний лист, розпочати новий лист і т. д., а виконання програми буде заблоковане допоки користувач не натисне клавішу, щоб обрати команду.

Натомість, у випадку інверсії управління, програма пишеться із використанням програмного каркасу, який знає загальні поведінкові і графічні елементи, такі як віконний інтерфейс, меню, керування мишкою тощо Користувацький код «заповнює пробіли» у каркасі, такі як надавання таблиці елементів меню і реєстрація підпрограм для кожного елемента, але відслідковування дій користувача і виклик пов'язаної підпрограми є завданням каркасу. У прикладі поштового клієнта, каркас може слідкувати за клавіатурою і мишкою і викликати команду обрану користувачем, також одночасно з цим моніторити мережевий інтерфейс, щоб помітити прибуття нового повідомлення і оновити екран коли з'являється мережева активність. Цей самий каркас можна використати як скелет для програми електронних таблиць або текстового редактора. З іншого боку каркас нічого не знає про веб-оглядачі, електронні таблиці, текстові редактори; втіленням їх функціональності займається користувацький код [34].

Spring Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java [33].

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної

моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB).

4.2.4 Фінальна архітектура

Клієнт додатку вводить свої дані у формі, які передаються використовуючи POST запит в контролер програми. При попаданні в контролер дані перевіряються на коректність і передаються в сервіс рекомендацій. Він відповідальний за отримання масиву кодів рекомендацій з регістра правил і збереження в базу даних. Збереження відбувається в базу Monog DB і може використовуватися в подальшому для оптимізації алгоритмів правил. Реєстратура правил зберігає список всіх класів, які успадковані від абстрактного класу «Правило» і поміщені в контейнер інверсії контролю. При попаданні метрик користувача в цей клас, реєстратура ітеративно застосовує метрики до всіх правил і трансформує їх у коді рекомендацій. Ці коди передаються назад у контролер, інтернаціоналізуються і виводяться на стороні клієнта за вказаними в json файлі правилами. Ця робота показана на рисунку 4.4.

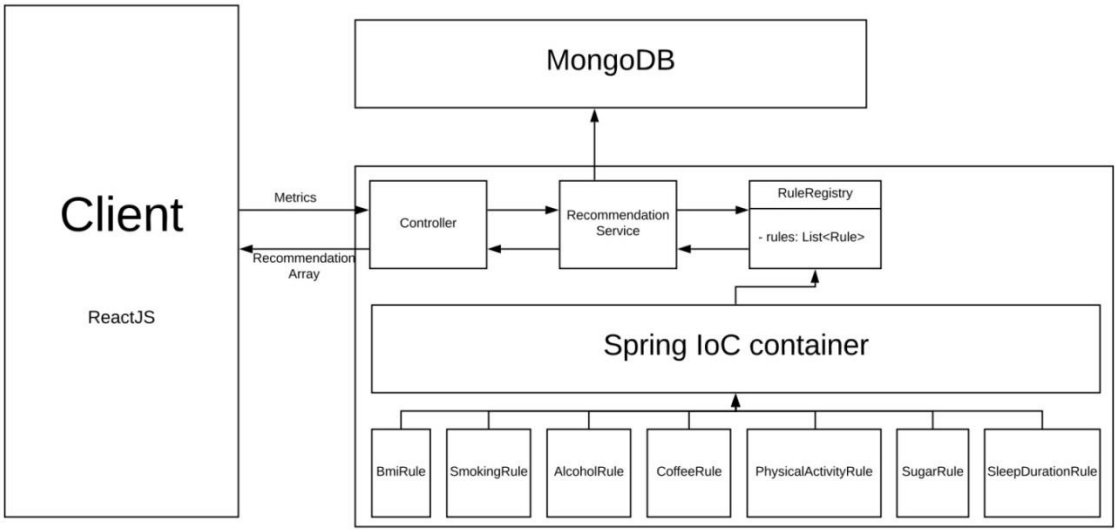

Рисунок 4.4 – Схема роботи програмного забезпечення

4.3 Вибір язика програмування

Kotlin — статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту [36].

Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.

Мова розробляється з 2010 року, публічно представлена в липні 2011. Сирцевий код було відкрито в лютому 2012. В лютому було випущено milestone 1, який містив плагін для IDEA. У червні — milestone 2 з підтримкою Android. У грудні 2012 року вийшов milestone 4 та забезпечив підтримку Java 7. Станом на листопад 2015 року основні можливості мови стабілізовані, готується реліз версії 1.0. В грудні 2015 року з'явився реліз-кандидат версії 1.0, а 15 лютого 2016 року відбувся реліз версії 1.0 [36].

З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки додатків для платформи Android.

З 7 травня 2019 року є рекомендованою мовою для розробки Android додатків.

Ведучий розвиток Андрій Бреслав заявив, що Котлін розроблений як об'єктно-орієнтована мова індустріальної мови і "краща мова", ніж Java, але все ще повністю взаємодіє з кодом Java, що дозволяє компаніям здійснювати поступову міграцію з Java на Kotlin. Точки з комою є необов'язковими як термінатор оператора; в більшості випадків достатньо нового рядка для компілятора, щоб зробити висновок про закінчення оператора. Декларації змінних Kotlin і списки параметрів мають тип даних після назви змінної (і з роздільниками двокрапкою), подібно до Pascal. Змінні в Kotlin можуть бути незмінними, оголошеними ключовим словом val, або mutable, оголошеними ключовим словом var.

Члени класу за замовчуванням є загальнодоступними, а самі класи є за замовчуванням остаточними, а це означає, що створення похідного класу буде вимкнено, якщо базовий клас не буде оголошено ключовим словом open. На додаток до класів і методів (які називаються функціями-членами в Котліні) об'єктно-орієнтованого програмування, Котлін також підтримує процедурне програмування з використанням функцій [36].

4.4 Результати розробки

У результаті розробки вийшов веб-додаток, який працює на будь-якому електронному пристрою, де є наявність інтернету та силка на сервер, де зберігається цей додаток.

Початок роботи розпочинається з того, що додаток автоматично заповняє текстові поля, такі як:

- Вага
- Зріст
- Часи сну
- Фізична активність
- Фізичний стан
- Куріння
- Вживання алкоголю
- Вживання цукру
- Вживання кофеїну

Приклад наведено на рисунку 4.5.

Приклад введення параметрів можно побачити на рисунку 4.6.

Після введення параметрів розпочинається аналіз введенної інформаціі та программа на виході надає оцінку здоров’я користувача, рекомендаціі та попереджає о можливих наслідках зі здоров’ям. Приклад таких рекомендацій можно побачити на рисунку 4.7.

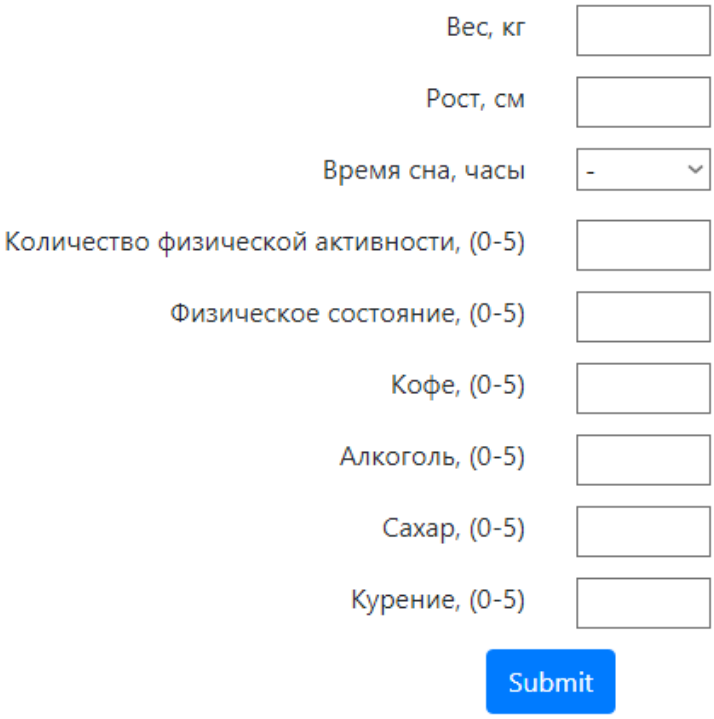

Рисунок 4.5 – Начальний екран роботи програми з текстовими віконцями

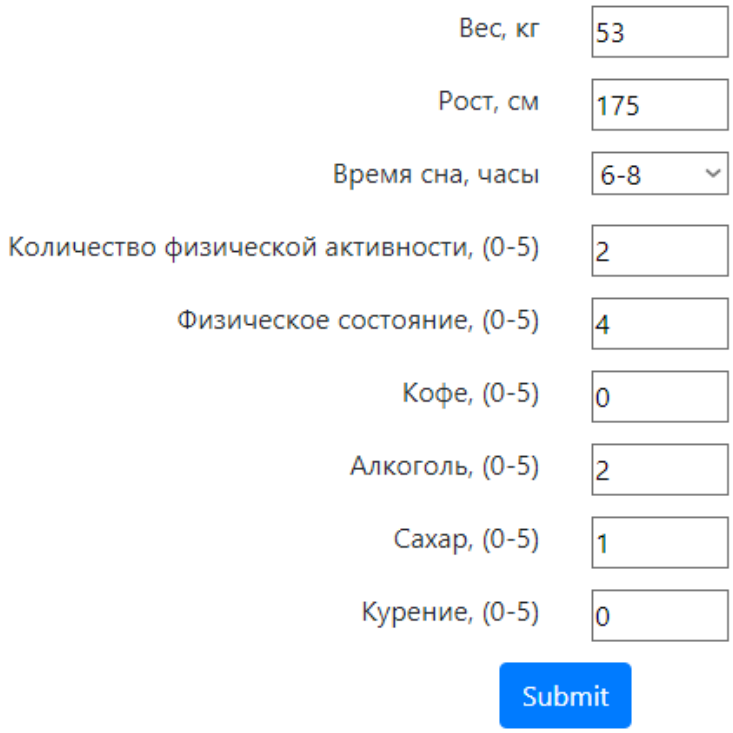

Рисунок 4.6 – Приклад заповнення текстових віконець

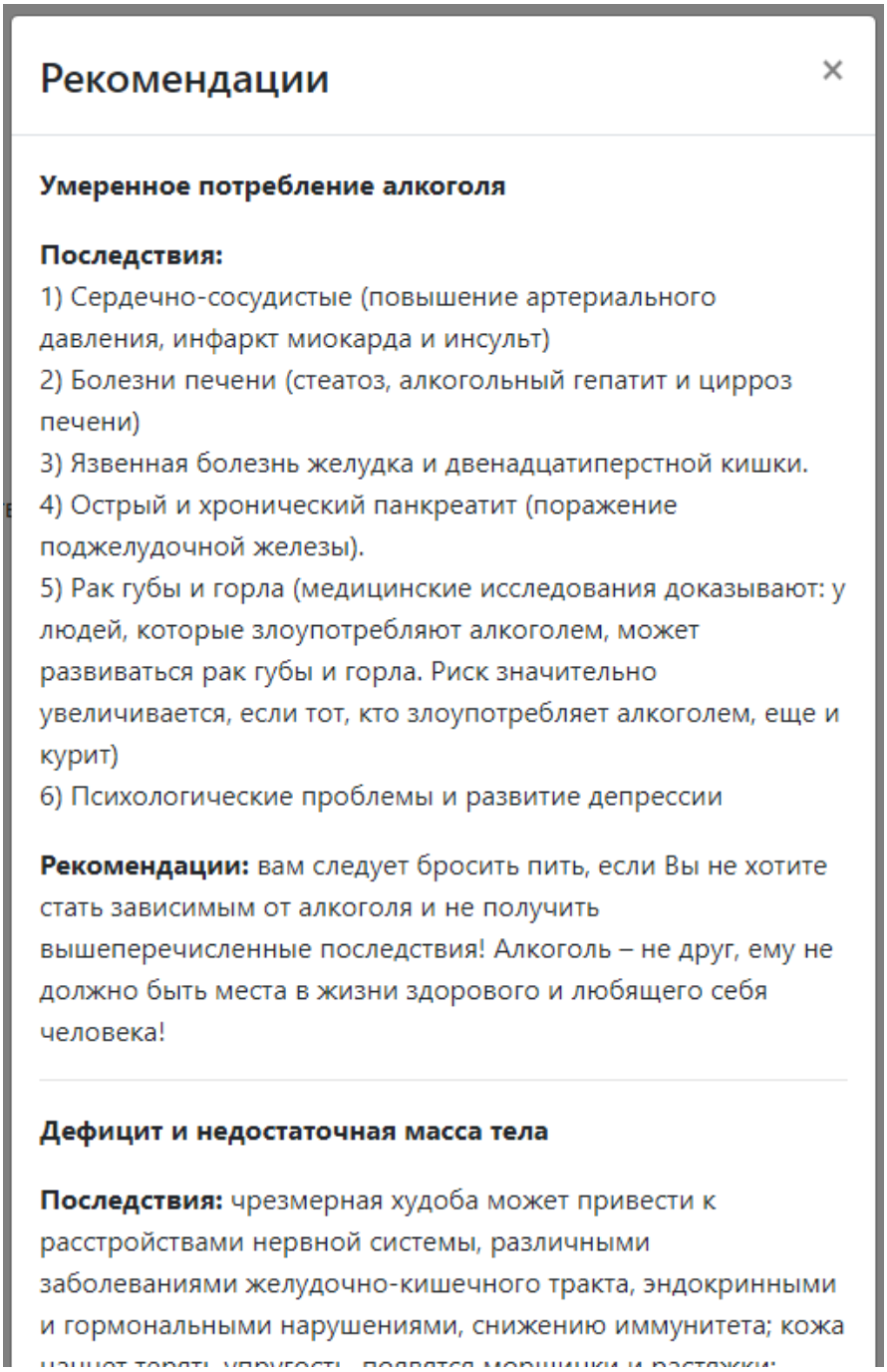

Рисунок 4.7 – Приклад рекомендацій, які надаються користувачеві

5 ЕКОНОМІЧНА ЧАСТИНА

Мета економічної частини дипломного проекту – розрахувати витрати на розробку програмного забезпечення і визначити економічну ефективність від його впровадження.

Програмний продукт (ПП) – поряд з апаратними засобами, найважливіша складова інформаційних технологій, що включає комп'ютерні програми і дані, призначені для вирішення певного кола задач і зберігаються на машинних носіях.

Програмний продукт являє собою або дані для використання в інших програмах, або алгоритм, реалізований у вигляді послідовності інструкцій для процесора.

Метою створення програмного продукту є отримання необхідного і достатнього системного комплексу якісних програмних ізделій при умові реалізації ефективного процесу розробки і супроводу [31].

5.1 Опис програмного продукту

У процесі роботи було створено програмний продукт для аналізу антропо-метричних параметрів людини. Цей додаток має наступні можливості:

а. Запис параметрів користувача.

б. Аналіз параметрів користувача.

в. Оцінка стану здоров'я користувача.

г. Видача рекомендацій по здоров'ю користувачеві.

д. Попередження користувача про наслідки.

е. Надання анонімності.

5.2 Розрахунок собівартості програмного продукту

Щоб оцінити вартість розроблюваного програмного продукту необхідно:

а. Скласти перелік робіт, які слід виконати.

б. Розрахувати трудовитрати на їх виконання.

в. Розрахувати заробітну плату розробників.

г. Розрахувати витрати на матеріали, що комплектує і машинний час.

У витрати на розробку програмного продукту також входять: вартість малоцінних і швидкозношуваних предметів, вартість оренди комп'ютера, відрахування з заробітної плати і т.д.

До переліку робіт, які необхідно виконати входить:

а. Формулювання постановки задачі.

б. Аналіз вимог, проектування функціональних і нефункціональних можливостей програмного продукту.

в. Розробка програмного продукту.

г. Впровадження продукту.

5.3 Виконавці роботи

Для управління ходом робіт і ведення всього проекту в цілому необхідна посада керівника. Для проектування підсистеми та її подальшого налагодження, і введення в експлуатацію необхідна участь програміста. Для визначення правильного функціонування програми необхідний тестувальник. Склад групи для розробки програмного продукту наведено у таблиці 5.1.

Таблиця 5.1 - Склад групи для розробки програмного продукту

Посади	Посадові оклади, грн	
	Місячні	Денні
Керівник	41600	1890,91
Програміст	67600	3072,72
Тестувальник	28600	1300

При нарахуванні заробітної плати за місяць приймаємо 22 робочих дня.

5.4 Перелік робіт для створення програмного продукту

Наведемо переліки робіт для розробників програмного продукту. Тривалість робіт встановлена дослідним шляхом.

Заробітна плата (ЗП) – це виражена в грошовій формі частина доходів, що надходить в особисте споживання працівників відповідно до кількості і якості витраченого ними праці.

Перелік робіт, що виконуються співробітниками і їх тривалість приведена в таблиці 5.2.

Таблиця 5.2 - Перелік робіт для створення програмного продукту

Номер стадії	Найменування стадій і етапів	Тривалість, Дні	Виконавці, кількість			Трудоємкість, люд/дні
			Керівник	Програміст	Тестувальник	
Розробка технічного завдання						
1	Організаційна підготовка до створення програмного продукту	2	2	0	0	2
2	Розробка постановки завдання	2	2	0	0	2
Разом:		**4**	**4**	**0**	**0**	**4**
Постановка задачі						
1	Розробка математичної моделі і алгоритмів	3	3	0	0	3
2	Структурування даних	3	3	0	0	3
3	Технічне забезпечення	2	1	1	0	2
4	Розробка опису завдання	2	2	0	0	2

Продовження таблиці 5.2

Разом:		**10**	**9**	**1**	**0**	**10**
Розробка ПП						
1	Розробка алгоритмів	4	0	4	0	4
2	Розробка інструкцій	2	0	1	1	2
3	Розробка програми	4	0	2	2	4
4	Проведення тестов	2	0	0	2	2
Разом:		**12**	**0**	**7**	**5**	**12**
Впровадження						
1	Тестування і попереднє випробування	3	3	3	3	9
2	Налагодження і корекція інструкції	2	2	2	2	6
Разом:		**5**	**5**	**5**	**5**	**15**
Всього:		**31**	**18**	**13**	**10**	**41**

Отримуємо, що трудомісткість роботи керівника, програміста і тестувальника становить 18, 13 і 10 відповідно.

Розрахунок собівартості робіт почнемо з розрахунку фонду основної заробітної плати (ОЗП) розробників, з урахуванням трудовитрат, кількості виконавців і середньоденної заробітної плати. Фонд основної заробітної плати визначається за формулою:

$$З_{о\boldsymbol{с}} = З_{кер} * Т_{кер} + З_{прог} * Т_{прог} + З_{те\boldsymbol{с}т} * Т_{те\boldsymbol{с}т} \quad (5.1)$$

де $Т_{кер}, Т_{прог}, Т_{те\boldsymbol{с}т}$ – трудомісткість роботи керівника, програміста і тестувальника;

$З_{кер}, З_{прог}, З_{те\boldsymbol{с}т}$ – заробітні плати керівника, програміста і тестувальника відповідно.

Звідси:

$З_{ос}$= 1890,91*18+ 3072,72*13 + 1300*10 = 86972,38 грн.

Нехай додаткова заробітна плата (ДЗП) визначена у розмірі 20% ОЗП.

Додаткова заробітна плата (ДЗП) – включає в себе компенсацію за час відпустки, оплату за перебування в декретній відпустці, лікарняні виплати, виплати за роботу неповнолітніх підлітків, за виконання громадських чи державних робіт, виплата допомоги при звільненні та інші виплати.

$$З_{дод} = З_{ос} * \frac{Н_{дод}}{100} \quad (5.2)$$

де $З_{ос}$ – основна заробітна плата;

$Н_{дод}$ – відсоток відрахувань до додаткової заробітної плати основних виробничих робочих (20%).

$$З_{дод} = 86972{,}38 * 0.2 = 17394{,}48 \text{ грн.}$$

У слідстві можемо порахувати фонд заробітної плати:

$$Ф_{зп} = З_{ос} + З_{дод}, \quad (5.3)$$

де $З_{ос}$ – основна заробітна плата;

$З_{дод}$ – додаткова заробітна плата.

Разом:

$$Ф_{зп} = З_{дод} + З_{ос} = 86972{,}38 + 17394{,}48 = 104366{,}85 \text{ грн.}$$

Нарахування на заробітну плату у відсотках від основної і додаткової заробітних плат (ЄСВ - єдиний соціальний внесок [Словник, с. 76]) становить 22% [32].

$$З_{соц} = Ф_{зп} * \frac{Н_{соц}}{100}, \quad (5.4)$$

де $Ф_{зп}$ – фонд заробітної платні;

$Н_{соц}$ – відсоток відрахувань до фонду соціального захисту населення (22%).

Звідси:

$$З_{соц} = 104366{,}85 * \frac{22}{100} = 20873{,}37 \text{ грн.}$$

5.5 Розрахунок кошторису і ціни на розробку програми

Розрахуємо витрати на матеріали і комплектуючі, необхідні для написання програми. Результати занесені в таблицю 5.3.

Таблиця 5.3 - Витрати на матеріали

№ п/п	Матеріал	Призначення	Кількість, шт	Ціна за одиницю, грн	Сума, грн
1	Папір формату А4 (упаковка 500 аркушів)	Оформлення документації та звітів	1	101,00	101,00
2	Картридж для принтера	Друк документації	1	460,00	460,00
3	Флеш – карта (16 Гб)	Збереження ПО	1	250,00	250,00
ВСЬОГО:					**811,00**

У підсумку загальна сума витрат на матеріали складає - 811,00 грн.

Вартість інтренет послуг за 29 днів склав 150 грн.

Для виконання робіт, пов'язаних з проектуванням програмного продукту й настройки, потрібні ПК з ліцензійною операційною системою Windows 10 (оперативна пам'ять не менше 4 Гб, дискова пам'ять не менше 1 Гб) вартістю 27413,00 грн., Принтер - 2123,00 грн. Результати занесені в таблицю 5.4

Таблиця 5.4 - Вартість основних засобів

№ п/п	Найменування	Кількість, шт.	Ціна за одиницю, грн	Сума, грн
1	Персональний комп'ютер з ліцензійною операційною системою Windows 10	1	27413,00	27413,00
2	Принтер	1	2123,00	2123,00
ВСЬОГО:				**29536,00**

Визначимо витрачений машинний час. Будемо вважати, що керівник користується комп'ютером в середньому 4 години за робочий день, а програміст і тестувальник, дотримуючись норм охорони праці, в середньому 6 годин.

$$Т_{мч} = ВИ_{кер} * Т_{кер} + ВИ_{прог} * Т_{прог} + ВИ_{тест} * Т_{тест} \quad (5.5)$$

де $Т_{кер}, Т_{прог}, Т_{тест}$ – трудомісткість роботи керівника, програміста і тестувальника;

$ВИ_{рук}, ВИ_{прог}, ВИ_{тест}$ – час роботи керівника, програміста і тестувальника за комп'ютером в середньому за день відповідно.

Отримаємо:

$Т_{мч} = 4 * 18 + 6 * 13 + 6 * 10 = 210$ годин.

Вартість години машинного часу $Ч_{мч}$ будемо вважати рівною 2грн.

$$C_{мч} = Т_{мч} * Ч_{мч}, \quad (5.6)$$

де $Т_{мч}$ – затраченное машинное время;

$Ч_{мч}$ – стоимость часа машинного времени (2 грн.).

Отримаємо:

$$C_{мч} = 210 * 2 = 420 \text{ грн.}$$

Амортизаційні відрахування (АМВ) – частина вартості основних засобів, що входять у вартість готової продукції.

Річна норма амортизаційних відрахувань (АМВ) розраховується як 25% від вартості одного комп'ютера і його комплектуючих.

$$АМВ = 27413{,}00 * 0{,}25 = 6853{,}25 \text{ грн.}$$

Норма амортизаційних відрахувань на період роботи проекту становить 29 робочих днів, розраховується за такою формулою:

$$АМВ_{на\ раб.період} = \frac{АМВ}{264} * 29 \quad (5.7)$$

$$АМВ_{на\ раб.період} = \frac{6853{,}25}{264} * 29 = 752{,}82 \text{ грн}$$

Собівартість – це вартісна оцінка використаних в процесі виробництва продукції (робіт, послуг) природних ресурсів, сировини, матеріалів, основних фондів, трудових ресурсів та інших витрат на її виробництво і реалізацію.

Собівартість розробки програмного продукту дорівнює сумі всіх вищезазначених витрат:

$$C_{\text{р}} = З_{\text{м}} + Ф_{\text{зп}} + З_{соц} + З_{\text{ов}} + C_{\text{мч}} + А\text{МВ}_{на\ раб.період}, \quad (5.9)$$

де $З_{\text{м}}$ – вартість витрат на матеріали;

$Ф_{\text{зп}}$ – фонд заработной платы;

$З_{соц}$ – єдиний соціальний фонд;

$C_{\text{мч}}$ – вартість електроенергії;

$А\text{МВ}_{на\ раб.період}$– амортизаційні відрахування;

$З_{\text{о}}$ – вартість витрат на обладання;

Звідси:

$$C_{\text{р}} = 961{,}00 + 104366{,}85 + 20873{,}37 + 420{,}00 + 752{,}82 + 2123{,}00 = 129497{,}04 \text{ грн.}$$

Таблиця 5.5 - Статті калькуляції на розробку програмного продукту

№, п/п	Стаття калькуляції	Витрати, грн.	Примітки
1	Матеріали	811,00	Таблиця 5.3
2	Основна заробітна плата	86972,38	$З_{ос} = З_{\text{кер}} * Т_{\text{кер}} + З_{\text{прог}} * Т_{\text{прог}} + З_{тест} * Т_{тест}$
3	Додаткова заробітна плата	17394,48	20% от п.2

Продовження таблиці 5.5

4	Інтернет послуги	150	Таблиця 5.3
5	ЄСВ - єдиний соціальний внесок	20873,37	22% от п.4
6	Обладання	2123,00	Таблиця 5.4
7	Використання електроенергії	420,00	$Т_{мч} = (ВИ_{кер} * Т_{кер} + ВИ_{прог} * Т_{прог} + ВИ_{тест} * Т_{тест}) * Ч_{мч}$
8	Амортизаційні відрахування на робочий період	752,82	$АМВ_{на\ раб.\ період} = \frac{З_{ос} * 0,25}{264} * 29$
9	Собівартість розробки	129497,04	Сума п.1 ÷ п.8

Висновок: на сьогоднішній день в умовах високої конкуренції на ІТ-ринку дуже важливо розробити програмний продукт за прийнятною для клієнта ціною. Для цього, до моменту завершення розробки програмного продукту, необхідно знати собівартість і ціну готової програми. Слід знати приблизну вартість готового продукту, щоб не найняти зайвих фахівців і не перевищити заплановану вартість програми, також необхідно скласти перелік і графік робіт, для того щоб укластися в терміни і не збільшити вартість продукту, шляхом збільшення терміну його розробки.

Внаслідок зазначених вище причин у даному розділі були проведені такі розрахунки:

- розрахунок собівартості розробленого програмного продукту склав 129497,04;
- розрахунок щоденної заробітної плати, для фахівців, що беруть участь в розробці ПП: керівник - 1890,91 грн, програміст - 3072,72 грн, тестувальник - 1300 грн;
- амортизаційні відрахування на робочий період складають 752,82грн.

ВИСНОВКИ

У даній роботі була розглянута система антропометричних параметрів і аналіз можливостей створення автоматизованої системи аналізу антропометричних параметрів.

На основі виконаної роботи можна зробити такі висновки:

1. Для більш точного аналізу стану здоров'я людини слід використовувати такі антропометричні дані:

• зріст

• вага

• стать

• наявність шкідливих звичок

• фізичну активність

• тривалість сну

2. Як модулі обробки і аналізу вибрати програму і формули, так як час роботи буде менше, а якість роботи краще.

3. Розробка системи аналізу антропометричних параметрів реальна і здійсненна.

На основі даного дослідження можна виділити такі перспективи:

1) систему можна пов'язати з ІІ, що уточнить прогнози здоров'я людини і більш точно оцінить якість життя людини

2) роботу над даною системою можна закріпити за командою професіоналів (DataScientist [Словник, с.76], Userexperiencedesigner [Словник, с.76], Frontenddeveloper [Словник, с.76], медперсонал, команда по роботі з антропометричними даними), що ще більш точно і поглиблено дозволить прогнозувати здоров'я людини і оцінювати його якість життя.

3) систему можна застосовувати в медицині, що поліпшить роботу медустанов і скоротить захворюваність і смертність людства

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бунак В. В. Антропометрия. Практический курс. — М., 1941 – 205 с.
2. Тегако Л. И., Марфина О. В. Практическая антропология. — Ростов-на-Дону, 2003 – 350 с.
3. Как быть здоровым (из зарубежного опыта обучения принципам здорового образа жизни). М.: Медицина, 1990 – 238 с.
4. Антропометрия // Большая советская энциклопедия: [в 30 т.] / гл. ред. А. М. Прохоров. — 3-е изд. — М.: Советская энциклопедия, 1969—1978 – 220 с.
5. Иван Репичев. Основные антропометрические данные человека [Електроний ресурс]: портал – режим доступа вільний: http://fb.ru/article/114288/osnovnyie-antropometricheskie-dannyie-cheloveka.
6. Антропометрия. [Електроний ресурс]: портал – Режим доступа вільний: https://dic.academic.ru/dic.nsf/ruwiki/1279850.
7. Журнал «Livescience». Приложение, прогнозирующее головную боль[Електроний ресурс]: портал – режим доступа вільний: https://www.livescience.com/health.
8. Статья на тему «Искусственный интеллект от Google прогнозирует дату смерти на 95%». [Електроний ресурс]: портал – режим доступа вільний: https://www.ukrinform.ru/rubric-technology/2484245-iskusstvennyj-intellekt-ot-google-prognoziruet-datu-smerti-na-95.html.
9. Tulu B. et al. SlipBuddy: A Mobile Health Intervention to Prevent Overeating //Proceedings of the 50th Hawaii International Conference on System Sciences. — 2017 – 300 с.
10. Фомин Н.А., Вавилов Ю.Н. Физиологические основы двигательной активности. - М.: Физкультура и спорт, 1991 - 224 с.
11. Индекс массы тела. [Електроний ресурс]: портал – режим доступа вільний: https://ru.wikipedia.org/wiki/Индекс_массы_тела
12. Акинщикова Г.И. Антропология: Учебное пособие. - Л.: Изд-во ЛГУ, 1974

– 345 с.

13. Дубровский В.И. Спортивная медицина. - М.:Владос, 1998 – 240 с.
14. Стаття на тему «Антропометрия.Общий уход за больными. Основные лечебно-диагностические процедуры». [Електроний ресурс]: портал – режим доступа вільний: http://www.bibliotekar.ru/449/2.htm.
15. Стаття на тему «Здоровый сон: сколько часов надо спать и чем опасно недосыпание». [Електроний ресурс]: портал – режим доступа вільний: https://med-expert.com.ua/journals/news/zdorovyj-son-skolko-chasov-nado-spat-chem-opasno-nedosypanie
16. Дембо А.Г. Врачебный контроль в спорте. - М.: Медицина, 1988 – 115 с.
17. Стаття на тему «Способ оценки резервов физического здоровья и работоспособности». [Електроний ресурс]: портал – режим доступа вільний: http://www.freepatent.ru/patents/2147208
18. «Handbook of Anthropometry», Timothy G. Lohman– 2017 – 447 с.
19. Preedy, Victor R. «Physical Measures of Human Form in Health and Disease» - 2011 – 500 с.
20. Стаття на тему «Оценка степени никотиновой зависимости студентов». [Електроний ресурс]: портал – режим доступа вільний: https://www.science-education.ru/ru/article/view?id=12079
21. «Bases of Human Factors Engineering/ Ergonomics», Kroemer, Karl H. E., Kroemer, Hiltrud J., Kroemer-Elbert, Katrin E. – 2010 – 310 с.
22. «Kroemer, Karl H. E., Kroemer, Hiltrud J., Kroemer-Elbert, Katrin E.», Stanley J. Ulijaszek, C. G. Nicholas Mascie-Taylor – 1994 – 169 с.
23. «Humanscale», Niels Diffrient, Elvin Tillie – 1974 – 400 с.
24. «Human Body Composition», Timothy G. Lohman – 1996 – 310 с.
25. «Anthropometrica: A Textbook of Body Measurement for Sports and Health Courses», Kevin Norton, Tim Olds – 1996 – 440 с.
26. «Anthropometric Standards: An Interactive Nutritional Reference of Body Size», A. Roberto Frisancho– 2008 – 276 с.
27. «Anthropometric Standardization Reference Manual», Timothy G. Lohman,

Алекс Ф.Рош, Reynaldo Martorell – 1988 – 238 с.

28. «Anthropometric Methods: Designing to Fit the Human Body», JohnArturRobak–1995 – 307 с.
29. «Anthropometry - Sports Physique Evaluation», Vijaya Lakshmi – 2005 – 198 с.
30. «Anthropometry and Physical Examination», Jay Webber Seaver– 1890 – 286 с.
31. Программное обеспечение. [Електроний ресурс]: портал – Режим доступа: http://www.tadviser.ru/index.php/Статья:Программное_обеспечение, - вільний.
32. Единый социальный взнос. [Електроний ресурс]: портал – Режим доступа: http://index.minfin.com.ua/index/social/, - вільний.
33. Spring Framework. [Електроний ресурс]: портал – Режим доступа: https://uk.wikipedia.org/wiki/Spring_Framework , - вільний.
34. Інверсія управління. [Електроний ресурс]: портал – Режим доступа: https://uk.wikipedia.org/wiki/Інверсія_управління , - вільний.
35. Клієнт-серверна архітектура. [Електроний ресурс]: портал – Режим доступа: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура , - вільний.
36. Kotlin. [Електроний ресурс]: портал – Режим доступа: https://en.wikipedia.org/wiki/Kotlin_(programming_language) , - вільний.
37. Модель-вид-контролер. [Електроний ресурс]: портал – Режим доступа: https://uk.wikipedia.org/wiki/Модель-вид-контролер , - вільний.
38. Статистика популярности смартфонов в Украине и почему это важно. [Електроний ресурс]: портал – Режим доступа:https://uip.me/2016/07/smartphones-in-ukraine/, - вільний.

ГЛОСАРІЙ

1. Єдиний соціальний внесок на загальнообов'язкове державне соціальне страхування - консолідований страховий внесок в Україні, збір якого здійснюється в системі загальнообов'язкового державного соціального страхування в обов'язковому порядку та на регулярній основі.

2. Краніометрія - методика вимірювання черепа, яка використовується з метою вивчення мінливості його будови.

3. Остеометрія - сукупність методів вимірювання кісток.

4. Соматометрія - сукупність методів, використовуваних для визначення розмірів і форми тіла людини.

5. Data Scientist - експерт з ШІ.

6. Frontend developer - експерт у створенні дизайну web-додатків, інтерфейсом сайтів, подачі інформацію в привабливому вигляді, приємному і зручному для користувачів.

7. User experience designer - експерт в проектуванні будь-яких призначених для користувача інтерфейсів в яких зручність використання так само важливо як і зовнішній вигляд.

8. Web-додаток - клієнт-серверний додаток, в якому клієнт взаємодіє з сервером за допомогою браузера, а за сервер відповідає веб-сервер. Логіка веб-додатки розподілена між сервером і клієнтом, зберігання параметрів здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є межплатформенному службами.

ДОДАТОК А

ПОЧАТКОВИЙ КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

build.gradle.kts

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

buildscript {
    repositories {
        mavenCentral()
    }
}

plugins {
    kotlin("jvm") version "1.4.20" apply false
    id("org.springframework.boot") version "2.4.0" apply false
    id("io.spring.dependency-management") version "1.0.10.RELEASE" apply false
    kotlin("plugin.spring") version "1.4.20" apply false
}

allprojects {
    group = "edu.khai"
    version = "0.0.1-SNAPSHOT"

    tasks.withType<Test> {
        useJUnitPlatform()
    }

    tasks.withType<KotlinCompile> {
        kotlinOptions {
            freeCompilerArgs = listOf("-Xjsr305=strict")
            jvmTarget = "1.8"
        }
    }
}

subprojects {
    repositories {
        mavenCentral()
    }
}
```

settings.gradle.kts

```
pluginManagement {
    repositories {
        gradlePluginPortal()
    }
    resolutionStrategy {
        eachPlugin {
            if (requested.id.id == "org.springframework.boot") {
                useModule("org.springframework.boot:spring-boot-gradle-plugin:${re-
quested.version}")
            }
        }
```

```
    }
}

rootProject.name = "health-recommendations-service"
include("commons", "recommendations-service", "test-data-producer", "device-data-aggre-
gator", "device-data-service")
```

docker-compose.yml

```
version: '2.1'

services:
  zoo:
    image: zookeeper:3.4.9
    hostname: zoo
    ports:
      - "2181:2181"
    environment:
      ZOO_MY_ID: 1
      ZOO_PORT: 2181
      ZOO_SERVERS: server.1=zoo:2888:3888
    volumes:
      - ./zk-single-kafka-single/zoo/data:/data
      - ./zk-single-kafka-single/zoo/datalog:/datalog

  kafka:
    image: confluentinc/cp-kafka:5.5.1
    hostname: kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_LISTENERS:
LISTENER_DOCKER_INTERNAL://kafka:19092,LISTENER_DOCKER_EXTERNAL://${DOCKER_HOST_IP:-
127.0.0.1}:9092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
      KAFKA_ZOOKEEPER_CONNECT: "zoo:2181"
      KAFKA_BROKER_ID: 1
      KAFKA_LOG4J_LOGGERS: "kafka.control-
ler=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
    volumes:
      - ./zk-single-kafka-single/kafka/data:/var/lib/kafka/data
    depends_on:
      - zoo
```

commons/build.gradle.kts

```
plugins {
    kotlin("jvm")
}

dependencies {
    implementation(kotlin("stdlib-jdk8"))
    implementation("javax.validation:validation-api:2.0.1.Final")
}
```

commonts/Metrics.kt

```
package edu.khai.healthrecommendationsservice.api

import javax.validation.constraints.Max
import javax.validation.constraints.Min
```

```
data class Metrics(
    @get:Min(0)
    val height: Double? = null,
    @get:Min(0)
    val weight: Double? = null,
    @get:Min(0)
    val sleepDuration: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val physicalFrequency: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val physicalState: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val smoking: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val alcohol: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val sugar: Int? = null,
    @get:Min(0)
    @get:Max(5)
    val coffee: Int? = null
)
```

commons/KafkaTopics.kt

```
package edu.khai.healthrecommendationsservice.commons

class KafkaTopics {
    companion object {
        const val DEVICE_DATA = "events_device_data_json_v1"
        const val DEVICE_DATA_AGGREGATED = "events_device_data_aggregated_json_v1"
    }
}
```

recommendations-service/build.gradle.kts

```
plugins {
    id("org.springframework.boot")
    id("io.spring.dependency-management")
    kotlin("jvm")
    kotlin("plugin.spring")
}

dependencies {
    implementation(project(":commons"))
    implementation(kotlin("stdlib-jdk8"))
    implementation(kotlin("reflect"))
    implementation("org.springframework.boot:spring-boot-starter-data-mongodb-reac-
tive")
    implementation("org.springframework.boot:spring-boot-starter-webflux")
    implementation("org.springframework.boot:spring-boot-starter-validation")
    implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
    implementation("de.flapdoodle.embed:de.flapdoodle.embed.mongo")
    runtimeOnly("org.springframework.boot:spring-boot-devtools")
    testImplementation("org.springframework.boot:spring-boot-starter-test") {
        exclude(group = "org.junit.vintage", module = "junit-vintage-engine")
        exclude(group = "junit", module = "junit")
```

```
    }
    testImplementation("io.projectreactor:reactor-test")
    testImplementation("io.mockk:mockk:1.9")
}
```

recommendations-service/application.properties

```
server.port=5000
```

recommendations-service/HealthRecommendationsServiceApplication.kt

```
package edu.khai.healthrecommendationsservice

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class HealthRecommendationsServiceApplication

fun main(args: Array<String>) {
    runApplication<HealthRecommendationsServiceApplication>(*args)
}
```

recommendations-service/CorsConfig.kt

```
package edu.khai.healthrecommendationsservice.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.web.cors.CorsConfiguration
import org.springframework.web.cors.reactive.CorsWebFilter
import org.springframework.web.cors.reactive.UrlBasedCorsConfigurationSource

@Configuration
class CorsConfig {

    @Bean
    fun corsFilter(): CorsWebFilter {
        val config = CorsConfiguration()
        config.applyPermitDefaultValues()
        config.addAllowedOrigin("*")
        config.addAllowedHeader("*")
        config.addAllowedMethod("*")
        val source = UrlBasedCorsConfigurationSource()
        source.registerCorsConfiguration("/**", config)
        return CorsWebFilter(source)
    }

}
```

recommendations-service/Extensions.kt

```
package edu.khai.healthrecommendationsservice.handler

import java.util.StringJoiner
import javax.validation.Validator


fun <T> T.validate(validator: Validator): Boolean {
    val constraintViolations = validator.validate(this)
    if (constraintViolations.isNotEmpty()) {
        val stringJoiner = StringJoiner(" ")
```

```
        constraintViolations.forEach { loginModelConstraintViolation ->
            stringJoiner
                .add(loginModelConstraintViolation.getPropertyPath().toString())
                .add(":")
                .add(loginModelConstraintViolation.getMessage())
        }
        throw RuntimeException(stringJoiner.toString())
    }
    return true
}
```

recommendations-service/RecommendationHandler.kt

```
package edu.khai.healthrecommendationsservice.handler

import edu.khai.healthrecommendationsservice.api.Metrics
import edu.khai.healthrecommendationsservice.service.RecommendationService
import org.springframework.http.MediaType
import org.springframework.stereotype.Component
import org.springframework.web.reactive.function.server.ServerRequest
import org.springframework.web.reactive.function.server.ServerResponse
import org.springframework.web.reactive.function.server.ServerResponse.notFound
import org.springframework.web.reactive.function.server.ServerResponse.ok
import reactor.core.publisher.Mono
import javax.validation.Validator

@Component
class RecommendationHandler constructor(
    private val recommendationService: RecommendationService,
    private val validator: Validator
) {
    fun getRecommendations(request: ServerRequest): Mono<ServerResponse> {
        return request.bodyToMono(Metrics::class.java)
            .filter { it.validate(validator) }
            .flatMap { recommendationService.getRecommendations(it).collectList() }
            .flatMap { ok().contentType(MediaType.APPLICATION_JSON).bodyValue(it) }
            .switchIfEmpty(notFound().build())
    }

}
```

recommendations-service/RecommendationResult.kt

```
package edu.khai.healthrecommendationsservice.model

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.data.annotation.Id
import org.springframework.data.mongodb.core.mapping.Document

@Document
data class RecommendationResult(
    @Id val id: String? = null,
    val metrics: Metrics,
    val recommendation: String
)
```

recommendations-service/RecommendationResultRepository.kt

```
package edu.khai.healthrecommendationsservice.repository

import edu.khai.healthrecommendationsservice.model.RecommendationResult
import org.springframework.data.repository.reactive.ReactiveCrudRepository
```

```
interface RecommendationResultRepository : ReactiveCrudRepository<RecommendationResult,
Long>
```

recommendations-service/RecommendationRouter.kt

```
package edu.khai.healthrecommendationsservice.router

import edu.khai.healthrecommendationsservice.handler.RecommendationHandler
import org.springframework.context.annotation.Bean
import org.springframework.stereotype.Component
import org.springframework.web.reactive.function.server.router

@Component
class RecommendationRouter {

    @Bean
    fun route(recommendationHandler: RecommendationHandler) = router {
        "/recommendation".nest {
            POST("", recommendationHandler::getRecommendations)
        }
    }

}
```

recommendations-service/AlcoholRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class AlcoholRule : Rule<Int?>() {
    override fun evaluateMetric(metrics: Metrics): Int? {
        return metrics.alcohol
    }

    override fun evaluateRecommendation(metric: Int?): String? {
        return when (metric) {
            null -> null
            0 -> "rule.alcohol.norm"
            in 1..2 -> "rule.alcohol.moderate"
            in 3..4 -> "rule.alcohol.high"
            else -> "rule.alcohol.extremely-high"
        }
    }
}
```

recommendations-service/BmiRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class BmiRule : Rule<Double?>() {
    override fun evaluateMetric(metrics: Metrics): Double? {
        return if (metrics.weight != null && metrics.height != null) {
            metrics.weight!! / Math.pow(metrics.height!! / 100, 2.0)
```

```
        } else {
            null
        }
    }

    override fun evaluateRecommendation(metric: Double?): String? {
        return when {
            metric == null -> null
            metric < 18.5 -> "rule.bmi.deficit"
            metric < 25 -> "rule.bmi.norm"
            metric < 30 -> "rule.bmi.excess"
            else -> "rule.bmi.obesity"
        }
    }
}
```

recommendations-service/CoffeeRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class CoffeeRule : Rule<Int?>() {
    override fun evaluateMetric(metrics: Metrics): Int? {
        return metrics.coffee
    }

    override fun evaluateRecommendation(metric: Int?): String? {
        return when (metric) {
            null -> null
            0 -> "rule.coffee.norm"
            in 1..2 -> "rule.coffee.moderate"
            in 3..4 -> "rule.coffee.high"
            else -> "rule.coffee.extremely-high"
        }
    }
}
```

recommendations-service/PhysicalActivityRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class PhysicalActivityRule : Rule<Pair<Int?, Int?>>() {
    override fun evaluateMetric(metrics: Metrics): Pair<Int?, Int?> {
        return metrics.physicalFrequency to metrics.physicalState
    }

    override fun evaluateRecommendation(metric: Pair<Int?, Int?>): String? {
        val (physicalFrequency, physicalState) = metric
        return when {
            physicalFrequency == null || physicalState == null -> null
            physicalFrequency in 0..2 && physicalState in 0..2 -> "rule.physical-activ-
ity.low-activity-bad-state"
```

```
            physicalFrequency in 0..2 && physicalState == 3 -> "rule.physical-activ-
ity.low-activity-norm-state"
            physicalFrequency in 0..2 && physicalState in 4..5 -> "rule.physical-activ-
ity.low-activity-good-state"
            physicalFrequency == 3 && physicalState in 0..2 -> "rule.physical-activ-
ity.norm-activity-bad-state"
            physicalFrequency == 3 && physicalState == 3 -> "rule.physical-activ-
ity.norm-activity-norm-state"
            physicalFrequency == 3 && physicalState in 4..5 -> "rule.physical-activ-
ity.norm-activity-good-state"
            physicalFrequency in 4..5 && physicalState in 0..2 -> "rule.physical-activ-
ity.high-activity-bad-state"
            physicalFrequency in 4..5 && physicalState == 3 -> "rule.physical-activ-
ity.high-activity-norm-state"
            else -> "rule.sleep-duration.high-activity-good-state"
        }
    }
}
```

recommendations-service/Rule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.beans.factory.annotation.Autowired

abstract class Rule<T> {

    @Autowired
    fun register(ruleRegistry: RuleRegistry) {
        ruleRegistry.register(this)
    }

    fun getRecommendation(metrics: Metrics): String? {
        return evaluateRecommendation(evaluateMetric(metrics))
    }

    protected abstract fun evaluateMetric(metrics: Metrics): T
    protected abstract fun evaluateRecommendation(metric: T): String?


}
```

recommendations-service/RuleRegistry.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class RuleRegistry {

    private val registry: MutableList<Rule<*>> = ArrayList()

    fun register(rule: Rule<*>) {
        registry.add(rule)
    }

    fun getRecommendations(metrics: Metrics): List<String> {
```

```
        return registry.mapNotNull { it.getRecommendation(metrics) }
    }
}
```

recommendations-service/SleepDurationRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class SleepDurationRule : Rule<Int?>() {
    override fun evaluateMetric(metrics: Metrics): Int? {
        return metrics.sleepDuration
    }

    override fun evaluateRecommendation(metric: Int?): String? {
        return when {
            metric == null -> null
            metric <= 5 -> "rule.sleep-duration.deficit"
            metric <= 8 -> "rule.sleep-duration.norm"
            else -> "rule.sleep-duration.excess"
        }
    }
}
```

recommendations-service/SmokingRules.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class SmokingRule : Rule<Int?>() {
    override fun evaluateMetric(metrics: Metrics): Int? {
        return metrics.smoking
    }

    override fun evaluateRecommendation(metric: Int?): String? {
        return when (metric) {
            null -> null
            0 -> "rule.smoking.norm"
            in 1..2 -> "rule.smoking.moderate"
            in 3..4 -> "rule.smoking.high"
            else -> "rule.smoking.extremely-high"
        }
    }
}
```

recommendations-service/SugarRule.kt

```
package edu.khai.healthrecommendationsservice.service.rule

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component

@Component
class SugarRule : Rule<Int?>() {
    override fun evaluateMetric(metrics: Metrics): Int? {
        return metrics.sugar
```

```
    }

    override fun evaluateRecommendation(metric: Int?): String? {
        return when (metric) {
            null -> null
            0 -> "rule.sugar.norm"
            in 1..2 -> "rule.sugar.moderate"
            in 3..4 -> "rule.sugar.high"
            else -> "rule.sugar.extremely-high"
        }
    }
}
```

recommendations-service/Extensions.kt

```
inline fun <reified T> T.logger(): Logger {
    return LoggerFactory.getLogger(T::class.java)
}

inline fun <reified T, K> T.logReactiveError(throwable: Throwable): Mono<K> {
    logger().error("Error in reactive pipeline", throwable)
    return Mono.empty()
}
```

recommendations-service/RecommendationService.kt

```
package edu.khai.healthrecommendationsservice.service

import edu.khai.healthrecommendationsservice.api.Metrics
import edu.khai.healthrecommendationsservice.model.RecommendationResult
import edu.khai.healthrecommendationsservice.repository.RecommendationResultRepository
import edu.khai.healthrecommendationsservice.service.rule.RuleRegistry
import org.springframework.stereotype.Service
import reactor.core.publisher.Flux
import reactor.core.publisher.Mono

@Service
class RecommendationService constructor(
    private val recommendationResultRepository: RecommendationResultRepository,
    private val ruleRegistry: RuleRegistry
) {

    fun getRecommendations(metrics: Metrics): Flux<String> {
        return Flux.merge(saveRecommendations(metrics))
            .map { it.recommendation }
    }

    private fun saveRecommendations(metrics: Metrics): List<Mono<RecommendationResult>>
{
        return ruleRegistry.getRecommendations(metrics)
            .map {
                RecommendationResult(
                    metrics = metrics,
                    recommendation = it
                )
            }
            .map {
                recommendationResultRepository.save(it)
                    .onErrorResume { err -> logReactiveError(err) }
            }
```

```
    }
}
```

test-data-producer/build.gradle.kts

```
plugins {
    id("org.springframework.boot")
    id("io.spring.dependency-management")
    kotlin("jvm")
    kotlin("plugin.spring")
}

dependencies {
    implementation(project(":commons"))
    implementation(kotlin("stdlib-jdk8"))
    implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
    implementation("org.apache.commons:commons-lang3:3.11")
    implementation("org.springframework.kafka:spring-kafka")
    implementation("org.springframework.boot:spring-boot-starter")
    runtimeOnly("org.springframework.boot:spring-boot-devtools")
    testImplementation("org.springframework.boot:spring-boot-starter-test") {
        exclude(group = "org.junit.vintage", module = "junit-vintage-engine")
        exclude(group = "junit", module = "junit")
    }
}
```

test-data-producer/application.properties

```
spring.kafka.bootstrap-servers=localhost:9092
spring.kafka.producer.key-serializer=org.apache.kafka.common.serializa-
tion.StringSerializer
spring.kafka.producer.value-serializer=org.springframework.kafka.support.serial-
izer.JsonSerializer
```

test-data-producer/TestDataProducer.kt

```
package edu.khai.healthrecommendationsservice.devicedataaggregator

import edu.khai.healthrecommendationsservice.commons.KafkaTopics.Compan-
ion.DEVICE_DATA
import org.springframework.boot.CommandLineRunner
import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
import org.springframework.kafka.core.KafkaTemplate
import java.lang.Thread.sleep

@SpringBootApplication
class TestDataProducer(
    val generator: RandomDataGenerator,
    val template: KafkaTemplate<String, Any>
) : CommandLineRunner {

    override fun run(vararg args: String?) {
        while (true) {
            val metrics = generator.generateRandomMetrics()
            template.send(DEVICE_DATA, "1", metrics)
            sleep(100)
        }
    }
}
```

```
fun main(args: Array<String>) {
    runApplication<TestDataProducer>(*args)
}
```

test-data-producer/RandomDataGenerator.kt

```
package edu.khai.healthrecommendationsservice.devicedataaggregator

import edu.khai.healthrecommendationsservice.api.Metrics
import org.apache.commons.lang3.RandomUtils
import org.springframework.stereotype.Component

@Component
class RandomDataGenerator {

    fun generateRandomMetrics(): Metrics {
        return Metrics(
            height = RandomUtils.nextDouble(50.0, 200.0),
            weight = RandomUtils.nextDouble(30.0, 150.0),
            sleepDuration = RandomUtils.nextInt(0, 24),
            physicalFrequency = RandomUtils.nextInt(0, 5)
        )
    }
}
```

device-data-aggregator/build.gradle.kts

```
plugins {
    id("org.springframework.boot")
    id("io.spring.dependency-management")
    kotlin("jvm")
    kotlin("plugin.spring")
}

dependencies {
    implementation(project(":commons"))
    implementation(kotlin("stdlib-jdk8"))
    implementation("org.apache.commons:commons-lang3:3.11")
    implementation("org.springframework.kafka:spring-kafka")
    implementation("org.apache.kafka:kafka-streams")
    implementation("org.springframework.boot:spring-boot-starter")
    runtimeOnly("org.springframework.boot:spring-boot-devtools")
    testImplementation("org.springframework.boot:spring-boot-starter-test") {
        exclude(group = "org.junit.vintage", module = "junit-vintage-engine")
        exclude(group = "junit", module = "junit")
    }
}
```

device-data-aggregator/application.properties

```
spring.kafka.streams.application-id=device-data-aggregator
spring.kafka.streams.bootstrap-servers=localhost:9092
spring.cloud.stream.kafka.streams.binder.configuration.de-
fault.key.serde=org.apache.kafka.common.serialization.Serdes$StringSerde
spring.cloud.stream.kafka.streams.binder.configuration.de-
fault.value.serde=org.apache.kafka.common.serialization.Serdes$StringSerde
```

device-data-aggregator/DeviceDataAggregator.kt

```
package edu.khai.healthrecommendationsservice.devicedataaggregator

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication
```

```
@SpringBootApplication
class DeviceDataAggregator

fun main(args: Array<String>) {
    runApplication<DeviceDataAggregator>(*args)
}
```

device-data-aggregator/KafkaStreamsConfig.kt

```
package edu.khai.healthrecommendationsservice.devicedataaggregator

import edu.khai.healthrecommendationsservice.api.Metrics
import edu.khai.healthrecommendationsservice.commons.KafkaTopics.Compan-
ion.DEVICE_DATA
import edu.khai.healthrecommendationsservice.commons.KafkaTopics.Compan-
ion.DEVICE_DATA_AGGREGATED
import org.apache.kafka.common.serialization.Serde
import org.apache.kafka.common.serialization.Serdes
import org.apache.kafka.streams.KeyValue
import org.apache.kafka.streams.StreamsBuilder
import org.apache.kafka.streams.kstream.Consumed
import org.apache.kafka.streams.kstream.KStream
import org.apache.kafka.streams.kstream.Materialized
import org.apache.kafka.streams.kstream.Produced
import org.apache.kafka.streams.kstream.TimeWindows
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.kafka.annotation.EnableKafka
import org.springframework.kafka.annotation.EnableKafkaStreams
import org.springframework.kafka.support.serializer.JsonDeserializer
import org.springframework.kafka.support.serializer.JsonSerializer
import java.time.Duration


@Configuration
@EnableKafka
@EnableKafkaStreams
class KafkaStreamsConfig {

    companion object {
        private val WINDOW_DURATION: Duration = Duration.ofMinutes(1)
    }

    @Bean
    fun aggregatedMetricsStream(kStreamBuilder: StreamsBuilder): KStream<String,
Metrics> {
        val stream = kStreamBuilder
            .stream(DEVICE_DATA, Consumed.with(Serdes.String(), metricsSerde()))
        val aggregatedStream = stream.groupByKey()
            .windowedBy(TimeWindows.of(WINDOW_DURATION))
            .aggregate(
                { MetricsSample() },
                { _, value, agg -> agg.add(value) },
                Materialized.with(Serdes.String(), metricsSampleSerde())
            )
            .mapValues(MetricsSample::reduce)
            .toStream()
            .map { key, value -> KeyValue.pair(key.key(), value) }
```

```
        aggregatedStream.to(DEVICE_DATA_AGGREGATED, Produced.with(Ser-
des.String(), metricsSerde()))
        return aggregatedStream
    }

    @Bean
    fun metricsSerde(): Serde<Metrics> {
        return Serdes.serdeFrom(JsonSerializer(), JsonDeserializer(Met-
rics::class.java))
    }

    @Bean
    fun metricsSampleSerde(): Serde<MetricsSample> {
        return Serdes.serdeFrom(JsonSerializer(), JsonDeserializer(Metrics-
Sample::class.java))
    }

}
```

device-data-aggregator/MetricsSample.kt

```
package edu.khai.healthrecommendationsservice.devicedataaggregator

import edu.khai.healthrecommendationsservice.api.Metrics
import kotlin.math.roundToInt

data class MetricsSample(
    val metricsList: MutableList<Metrics> = mutableListOf()
) {

    fun add(metrics: Metrics): MetricsSample {
        metricsList.add(metrics)
        return this
    }

    fun reduce(): Metrics = Metrics(
        height = metricsList.averageOrNull { it.height },
        weight = metricsList.averageOrNull { it.weight },
        sleepDuration = metricsList.roundedAverageOrNull { it.sleepDuration },
        physicalFrequency = metricsList.roundedAverageOrNull { it.physicalFre-
quency },
        physicalState = metricsList.roundedAverageOrNull { it.physicalState },
        smoking = metricsList.roundedAverageOrNull { it.smoking },
        alcohol = metricsList.roundedAverageOrNull { it.alcohol },
        sugar = metricsList.roundedAverageOrNull { it.sugar },
        coffee = metricsList.roundedAverageOrNull { it.coffee }
    )
}

private fun <T> Collection<T>.averageOrNull(transform: (T) -> Double?): Double?
=
    this.mapNotNull(transform).averageOrNull()

@JvmName("averageOrNullOfDouble")
private fun Collection<Double>.averageOrNull(): Double? = if (this.isEmpty())
null else this.average()

private fun <T> Collection<T>.roundedAverageOrNull(transform: (T) -> Int?): Int?
=
```

```
    this.mapNotNull(transform).averageOrNull()?.roundToInt()

@JvmName("averageOrNullOfInt")
private fun Collection<Int>.averageOrNull(): Double? = if (this.isEmpty()) null
else this.average()
```

device-data-service/build.gradle.kts

```
plugins {
    id("org.springframework.boot")
    id("io.spring.dependency-management")
    kotlin("jvm")
    kotlin("plugin.spring")
}

dependencies {
    implementation(project(":commons"))
    implementation(kotlin("stdlib-jdk8"))
    implementation(kotlin("reflect"))
    implementation("org.springframework.boot:spring-boot-starter-webflux")
    implementation("org.springframework.kafka:spring-kafka")
    implementation("org.apache.kafka:kafka-streams")
    implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
    runtimeOnly("org.springframework.boot:spring-boot-devtools")
    testImplementation("org.springframework.boot:spring-boot-starter-test") {
        exclude(group = "org.junit.vintage", module = "junit-vintage-engine")
        exclude(group = "junit", module = "junit")
    }
    testImplementation("io.projectreactor:reactor-test")
}
```

device-data-service/application.properties

```
server.port=5001
spring.kafka.streams.application-id=device-data-service
spring.kafka.streams.bootstrap-servers=localhost:9092
spring.cloud.stream.kafka.streams.binder.configuration.de-
fault.key.serde=org.apache.kafka.common.serialization.Serdes$StringSerde
spring.cloud.stream.kafka.streams.binder.configuration.de-
fault.value.serde=org.apache.kafka.common.serialization.Serdes$StringSerde
```

device-data-service/DeviceDataSerivceApplication.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class DeviceDataServiceApplication

fun main(args: Array<String>) {
    runApplication<DeviceDataServiceApplication>(*args)
}
```

device-data-service/RecommendationServiceClient.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.client

import edu.khai.healthrecommendationsservice.api.Metrics
import org.springframework.stereotype.Component
import org.springframework.web.reactive.function.BodyInserters
import org.springframework.web.reactive.function.client.WebClient
```

```
import reactor.core.publisher.Flux

@Component
class RecommendationServiceClient(private val recommendationService: WebClient)
{

    fun getRecommendations(metrics: Metrics): Flux<String> {
        return recommendationService.post()
            .uri("recommendation")
            .body(BodyInserters.fromValue(metrics))
            .retrieve()
            .bodyToFlux(String::class.java)
    }

}
```

device-data-service/ClientsConfig.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.web.reactive.function.client.WebClient

@Configuration
class ClientsConfig {

    @Bean
    fun recommendationService(builder: WebClient.Builder): WebClient {
        return builder.baseUrl("http://localhost:5000/").build()
    }
}
```

device-data-service/CorsConfig.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.config

import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.web.cors.CorsConfiguration
import org.springframework.web.cors.reactive.CorsWebFilter
import org.springframework.web.cors.reactive.UrlBasedCorsConfigurationSource

@Configuration
class CorsConfig {

    @Bean
    fun corsFilter(): CorsWebFilter {
        val config = CorsConfiguration()
        config.applyPermitDefaultValues()
        config.addAllowedOrigin("*")
        config.addAllowedHeader("*")
        config.addAllowedMethod("*")
        val source = UrlBasedCorsConfigurationSource()
        source.registerCorsConfiguration("/**", config)
        return CorsWebFilter(source)
    }

}
```

device-data-service/KafkaStreamsConfig.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.config

import edu.khai.healthrecommendationsservice.api.Metrics
import edu.khai.healthrecommendationsservice.commons.KafkaTopics
import org.apache.kafka.common.serialization.Serde
import org.apache.kafka.common.serialization.Serdes
import org.apache.kafka.streams.KafkaStreams
import org.apache.kafka.streams.StoreQueryParameters
import org.apache.kafka.streams.StreamsBuilder
import org.apache.kafka.streams.Topology
import org.apache.kafka.streams.kstream.Consumed
import org.apache.kafka.streams.kstream.KTable
import org.apache.kafka.streams.kstream.Materialized
import org.apache.kafka.streams.state.QueryableStoreTypes
import org.apache.kafka.streams.state.ReadOnlyKeyValueStore
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import org.springframework.kafka.annotation.EnableKafka
import org.springframework.kafka.annotation.EnableKafkaStreams
import org.springframework.kafka.config.StreamsBuilderFactoryBean
import org.springframework.kafka.support.serializer.JsonDeserializer
import org.springframework.kafka.support.serializer.JsonSerializer
import java.util.concurrent.CountDownLatch
import java.util.concurrent.TimeUnit


@Configuration
@EnableKafka
@EnableKafkaStreams
class KafkaStreamsConfig(val streamsBuilder: StreamsBuilder) {

    @Bean
    fun deviceDataTable(): KTable<String, Metrics> {
        return streamsBuilder.table(
            KafkaTopics.DEVICE_DATA_AGGREGATED,
            Consumed.with(Serdes.String(), metricsSerde())
                .withOffsetResetPolicy(Topology.AutoOffsetReset.EARLIEST),
            Materialized.`as`("device-data-aggregated")
        )
    }

    @Bean
    fun metricsSerde(): Serde<Metrics> {
        return Serdes.serdeFrom(JsonSerializer(), JsonDeserializer(Met-
rics::class.java))
    }

    @Bean
    fun latch(streamsBuilderFB: StreamsBuilderFactoryBean): CountDownLatch {
        val latch = CountDownLatch(1)
        streamsBuilderFB.setStateListener { newState, _ ->
            if (newState == KafkaStreams.State.RUNNING) {
                latch.countDown()
            }
        }
        return latch
    }
```

```
    @Bean
    fun deviceDataMaterializedTable(
        streamsBuilderFB: StreamsBuilderFactoryBean,
        deviceDataTable: KTable<String, Metrics>
    ): ReadOnlyKeyValueStore<String, Metrics> {
        streamsBuilderFB.start()
        latch(streamsBuilderFB).await(50, TimeUnit.SECONDS)
        return streamsBuilderFB.kafkaStreams.store(
            StoreQueryParameters.fromNameAndType(
                deviceDataTable.queryableStoreName(),
                QueryableStoreTypes.keyValueStore()
            )
        )
    }
}
```

device-data-service/DeviceDataHandler.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.handler

import edu.khai.healthrecommendationsservice.devicedataservice.service.DeviceDa-
taService
import org.springframework.http.MediaType
import org.springframework.stereotype.Component
import org.springframework.web.reactive.function.server.ServerRequest
import org.springframework.web.reactive.function.server.ServerResponse
import org.springframework.web.reactive.function.server.ServerResponse.notFound
import org.springframework.web.reactive.function.server.ServerResponse.ok
import reactor.core.publisher.Mono

@Component
class DeviceDataHandler constructor(
    private val deviceDataService: DeviceDataService,
) {
    fun getRecommendations(request: ServerRequest): Mono<ServerResponse> {
        return deviceDataService.getRecommendationsFromLatestDeviceData()
            .collectList()
            .flatMap { ok().contentType(Medi-
aType.APPLICATION_JSON).bodyValue(it) }
            .switchIfEmpty(notFound().build())
    }

}
```

device-data-service/DeviceDataRepository.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.repository

import edu.khai.healthrecommendationsservice.api.Metrics

interface DeviceDataRepository {

    fun findByKey(key: String): Metrics
}
```

device-data-service/KTableDeviceDataRepository.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.repository

import edu.khai.healthrecommendationsservice.api.Metrics
import org.apache.kafka.streams.state.ReadOnlyKeyValueStore
```

```
import org.springframework.stereotype.Repository

@Repository
class KTableDeviceDataRepository(val table: ReadOnlyKeyValueStore<String, Met-
rics>) : DeviceDataRepository {

    override fun findByKey(key: String): Metrics = table.get(key)

}
```

device-data-service/DeviceDataRouter.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.router

import edu.khai.healthrecommendationsservice.devicedataservice.handler.Device-
DataHandler
import org.springframework.context.annotation.Bean
import org.springframework.stereotype.Component
import org.springframework.web.reactive.function.server.router

@Component
class DeviceDataRouter {

    @Bean
    fun route(deviceDataHandler: DeviceDataHandler) = router {
        "/recommendation".nest {
            GET("", deviceDataHandler::getRecommendations)
        }
    }

}
```

device-data-service/DeviceDataService.kt

```
package edu.khai.healthrecommendationsservice.devicedataservice.service

import edu.khai.healthrecommendationsservice.devicedataservice.client.Recommen-
dationServiceClient
import edu.khai.healthrecommendationsservice.devicedataservice.repository.De-
viceDataRepository
import org.springframework.stereotype.Service
import reactor.core.publisher.Flux
import reactor.core.publisher.Mono

@Service
class DeviceDataService constructor(
    private val deviceDataRepository: DeviceDataRepository,
    private val client: RecommendationServiceClient,
) {

    fun getRecommendationsFromLatestDeviceData(): Flux<String> {
        return Mono.fromCallable { deviceDataRepository.findByKey("1") }
            .flatMapMany { client.getRecommendations(it) }
    }
}
```